# Analyse verteilter Systeme mit Hilfe von Prozeßautomaten

Von der Fakultät für Mathematik, Naturwissenschaften und Informatik
der Brandenburgischen Technischen Universität Cottbus

zur Erlangung des akademischen Grades

**Doktor der Naturwissenschaften**
**(Dr. rer. nat.)**

genehmigte Dissertation

vorgelegt von

Diplom Informatiker

**Peter H. Deussen**

geboren am 9. Dezember 1962 in Wolfenbüttel

Gutachter: Prof. Dr.-Ing. Monika Heiner

Gutachter: Prof. Dr. rer. nat. Peter H. Starke

Gutachter: Prof. Dr. rer. nat. Jörg Desel

Tag der mündlichen Prüfung: 7. Dezember 2001

Peter H. Deussen

# Analyse verteilter Systeme mit Hilfe von Prozeßautomaten

Dissertation

1999–2001

Fraunhofer Forschungsinstitut für offene Kommunikationssystem
Kaiserin-Augusta-Allee 31
10589 Berlin

Email: deussen@fokus.fhg.de, PDeussen@gmx.de

**Zusammenfassung**

Diese Arbeit behandelt einen Ansatz zur Verifikation verteilter Systeme. Zur Beschreibung des Verhaltens solcher Systeme verwenden wir die sog. *Semiwörter*, die eine spezielle Form von *pomsets* darstellen, sowie *Mazurkiewiczspuren*. Wir weisen nach, daß jedes nebenläufige System (d. h. ein System, für dessen Aktionen eine Unabhängigkeitsrelation angegeben werden kann) als verteiltes System aufgefaßt werden kann.

Wir beschreiben eine endliche Repräsentation des Verhaltens nebenläufiger und verteilter Systeme, die auf dem Begriff des *Prozeßautomaten* beruht, und geben Algorithmen zur Konstruktion derartiger Prozeßautomaten an.

Schließlich definieren wir die verteilte, agentenbasierte temporale Logik DCTL. Die Beschreibung eines Modelcheckers für diese Logik, der auf Prozeßautomaten operiert, schließt die Arbeit ab.

**Summary**

This thesis addresses the verification of distributed systems. The description of the behaviour of distributed systems is done by so-called semi words. Semi words are a dedicated type of *pomsets*. Also, *Mazurkiewicz traces* are used as a behavioural description. We will prove that every concurrent system (i. e. a system for that an independence relation can be defined on system actions) can be considered as a distributed system.

We describe a finite representation of the behaviour of concurrent and distributed systems based on the notion of *process automata*. We define algorithms for the construction of process automata.

Finally, we describe the distributed, agent based temporal logic DCTL. A model checking algorithm for DCTL is given. This model checker operates on process automata.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Algorithmenverzeichnis

# Einleitung

Nebenläufigkeit und Verteiltheit werden mit der Verlagerung informationstechnischer Vorgänge von lokalen auf globale Strukturen (LANs, WANs, mobile Netzwerke und — natürlich — das WWW) bei der Spezifikation und Validation von informationstechnischen Systemen zu primären Themen. Unter *Systemvalidation* wollen wir hierbei jede Maßname verstehen, die dazu beiträgt, daß ein informationstechnisches System einer gegebenen Spezifikation genügt, z. B. Tests und ein von allgemeinen Richtlinien gelenkter Entwicklungsprozeß, aber insbesondere auch *Systemverifikation*, d. h. der Nachweis (im Sinne eines mathematischen Beweises) der Korrektheit des Systems bzgl. seiner Spezifikation. Während jedoch Programmspezifikation und -verifikation in der Vergangenheit unter der Voraussetzung eines deterministischen und sequentiellen Programmablaufs erfolgen konnten (siehe [**48**] für eine gute Zusammenfassung klassischer Techniken), sehen wir uns heute mit der Notwendigkeit konfrontiert, Prozesse zu verstehen und zu beschreiben, die inhärent nichtdeterministisch sind und zudem teilweise unabhängige (d. h. nebenläufige) Teilprozesse umfassen.

Diese Arbeit behandelt die Verifikation verteilter Systeme, also solcher Systeme, die aus verschiedenen Komponenten besteht, deren Entwicklung bis auf definierte Interaktionen unabhängig voneinander verläuft und die auf einer räumlich verteilten Plattform realisiert sind. Beispiele für solche Systeme sind Client-Server-Architekturen, Kommunikationsprotokolle, verteilte Steuerungen industrieller Produktionsprozesse oder Multiprozessorarchitekturen. Die gängigen Entwurfsmethoden für solche Systeme berücksichtigen ihren verteilten Charakter bereits zumindest auf der syntaktischen Ebene, indem Komponenten und Kausalstrukturen zwischen Komponenten spezifiziert werden. Zum Beispiel sind *message sequence charts* (MSCs) [**49**] ein im Entwurf von Telekommunikationssystemen weit verbreiteter Formalismus, der explizit die Beschreibung derartiger Kausalbeziehungen erlaubt, während verschiedene Diagrammtypen der *unified modelling language* (UML) [**11**] ebenfalls die kausalbasierte Beschreibung verteilter Systeme unterstützen. Klassische Ansätze sind Milners *calculus of concurrent systems* (CCS) [**57**] oder Hoares *communicating sequential processes* (CSP) [**39**].

**Nebenläufige Systeme und ihre Halbordnungssemantik.** Diese Arbeit verwendet Transitionssysteme als allgemeines Systemmodell. Ein Transitionssystem ist durch eine Menge von Systemzuständen und eine Menge von Systemaktionen gekennzeichnet, die auf Systemzuständen operieren. Eine Aktion kann bei einem Zustand konzessioniert sein und in diesem Fall das System in einen anderen Zustand überführen. Nebenläufigkeit wird mit Hilfe einer Unabhängigkeitsrelation charakterisiert: Sind $a$ und $b$ unabhängige Systemaktionen, die bei einem Systemzustand simultan konzessioniert sind, so finden diese nebenläufig zueinander statt. Ein Transitionssystem zusammen mit einer Unabhängigkeitsrelation nennen wir ein Spursystem. Einer Reihe von Beschreibungsformen für verteilte Systeme läßt sich kanonisch ein Spursystem zuordnen. Eine wesentliche Einschränkung dieser Arbeit ist die Beschränkung auf endliche Transition- und Spursysteme.

Die klassische Semantik für Transitionssysteme basiert auf dem Sprachbegriff der Automatentheorie: Ein Transitionssystem bezeichnet eine Menge von Sequenzen von Systemaktionen, deren Hintereinanderausführung möglich ist; eine solche *sequentielle Semantik* berücksichtigt aber keine Abhängigkeits- bzw. Unabhängigkeitsbeziehungen zwischen Systemaktionen. Deshalb finden zunehmend Ansätze Beachtung, die solche Beziehungen konzeptionell umsetzen. Ein sehr prominenter Vertreter der Familie solcher *true concurrency semantics* sind die von Mazurkiewicz [**50, 51**] eingeführten *Spuren*, die heute als *Mazurkiewiczspuren* bezeichnet werden. Spuren sind Mengen von Sequenzen, die sich lediglich in der Reihenfolge unabhängiger Aktionen unterscheiden und deshalb als äquivalent angesehen werden. Die zugrundeliegende Idee ist, daß unabhängige Aktionen in beliebiger Reihenfolge stattfinden können. Die Kausalstruktur des unterliegenden nebenläufigen Systems ist in Mazurkiewiczsspuren also nur implizit repräsentiert. Wollen wir Nebenläufigkeit und Kausalität explizit sichtbar machen, können wir die Menge der möglichen Ereignisse eines Systems partiell ordnen: Sind $e_1$ und $e_2$ etwa Systemereignisse, so schreiben wir $e_1 < e_2$, wenn das Stattfinden von $e_1$ Voraussetzung für das Stattfinden von $e_2$ ist. Unter dieser Interpretation beobachten wir:

(a) Kein Ereignis kann seine eigene Voraussetzung sein, d. h. $<$ ist irreflexiv;
(b) ist $e_1$ Voraussetzung von $e_2$, so kann $e_2$ unmöglich $e_1$ verursacht haben, d. h. $<$ ist antisymmetrisch;
(c) wird $e_2$ von $e_1$ und $e_3$ von $e_2$ verursacht, so ist $e_1$ auch Voraussetzung von $e_3$, d. h. $<$ ist transitiv.

Die Ordnung $<$ ist also eine Halbordnung. Ereignisse, die nicht in einer solchen Ursache-Wirkungsbeziehung stehen ($e_1 \not< e_2$ und $e_2 \not< e_1$ für $e_1 \neq e_2$) werden als nebenläufig interpretiert.

Man beachte, daß sich die obige Argumentation auf Systemereignisse und nicht auf Systemaktionen bezieht; unter einem Systemereignis verstehen wir das Stattfinden einer Systemaktion. Um den Zusammenhang zwischen einem Ereignis und

der zugehörigen Aktion herzustellen, betrachten wir in dieser Arbeit solche Halbordnungen, die mit einer Beschriftungsfunktion versehen sind, die Systemereignisse auf Systemaktionen abbilden.

*Prim-Ereignisstrukturen*[1] [**59**] sind solche *beschrifteten Halbordnungen*. Die Abstraktion von konkreten Elementen zur Darstellung von Systemereignissen wird in Halbwörtern [**35**] (in [**67**] als *partial ordered multisets* (*Pomsets*) bezeichnet; dieser Begriff ist heute allgemein üblich) vollzogen, indem Isomorphieklassen von beschrifteten Halbordnungen betrachtet werden.[2] In [**78**] werden Halbwörter betrachtet, die keine Selbstnebenläufigkeiten enthalten; die resultierenden Äquivalenzklassen beschrifteter Halbordnungen werden als Semiwörter bezeichnet. Wir verwenden in dieser Arbeit eine besondere Ausprägung von Semiwörtern als Halbordnungssemantik für Spursysteme, nämlich solche Semiwörter, die konsistent zu der mit einem Spursystem assoziierten Unabhängigkeitsrelation sind. Wie sich zeigt, ergibt sich für solche Semiwörter eine interessante und nützliche Beziehung zu den Mazurkiewiczspuren: Es ist möglich, solche konsistenten Semiwörter bijektiv auf Mazurkiewiczsspuren abzubilden, d. h. unabhängigkeits-konsistente Semiwörter und Mazurkiewiczspuren sind verschiedene Repräsentationen desselben mathematischen Objekts.

Die genannten Semantiken nebenläufiger Systeme können als Semantik für Petri-Netze [**58, 66, 73, 81**], das in dieser Arbeit zur Notation von Beispielen und als Argumentationshilfe favorisierte Beschreibungsmittel für nebenläufige Systeme verwendet werden (tatsächlich enthalten die meisten der aufgeführten Referenzen einen expliziten Verweis auf Petri-Netze), sind aber keineswegs darauf beschränkt. Für Petri-Netze existiert noch eine weitere Semantik, die nicht auf andere Formalismen übertragen werden kann (zumindest nicht, ohne diese Formalismen in irgendeiner Weise als Petri-Netze aufzufassen): *Prozesse* bzw. *Abläufe* [**9**]. Solche Abläufe sind ihrerseits eingeschränkte Petri-Netze, sog. *Kausalnetze*.

Wir wollen in dieser Arbeit nichts zu der Frage beitragen, ob eine halbordnungsbasierte Semantik etwas zum Verständnis nebenläufiger Systeme beiträgt oder nicht — eine (teilweise amüsante) Diskussion dieses Themas findet sich in [**68**]. Der Author ist der Ansicht, daß im Gegensatz zu Sequenzen Halbordnungen zumindest eine intuitivere Darstellung nebenläufigen Systemverhaltens erlauben.

[1] Eine Ereignisstruktur ist eine beschriftete Halbordnung zusammen mit einer Konfliktrelation auf Ereignissen. Befinden sich zwei Ereignisse in Konflikt zueinander, so können sie nicht gemeinsam in derselben Verhaltensweise eines zugrundeliegenden Systems auftreten. Eine Ereignisstruktur beschreiben also mehrere alternative Verhaltensweisen. Ereignisstrukturen, die genau eine Systemverhaltensweise beschreiben, deren Konfliktrelation also leer ist, werden als Prim-Ereignisstrukturen bezeichnet.

[2] Die Arbeit mit Prim-Ereignisstrukturen wird vielfach durch Isomorphiefragen erschwert: Zwei isomorphe Prim-Ereignisstrukturen beschreiben, obwohl unterschiedlich, offenbar dasselbe Systemverhalten. Aus diesem Grund verwenden wir in dieser Arbeit Halbwörter, wobei wir großen Wert auf eine saubere Darstellung der Isomorphiebeziehung ihrer Repräsentanten legen.

Darüberhinaus kann durch die implizite oder explizite Verwendung einer Halbordnungssemantik das sog. *Zustandsexplosionsproblem* teilweise vermieden werden, so daß automatische Analysen überhaupt erst möglich werden.

**Verteilte Systeme.** Petri-Netze stellen keinen Komponentenbegriff und keine Kompositionsoperation zur Verfügung. Zwar existieren verschiedene Arbeiten, die Möglichkeiten zur Definition solcher Strukturierungsmittel behandeln (s. etwa [**7, 8, 45**]), wir machen in dieser Arbeit jedoch (ausser in Beispielen) keinen Gebrauch von einem Komponentenbegriff, der in der Syntax des Beschreibungsformalismus für verteilte Systeme verankert ist; auch für Spursysteme wird kein wie auch immer definiter Kompositionsoperator eingeführt.

Allerdings wird Nebenläufigkeit i. d. R. durch Verteilung realisiert. Sind also $a$ und $b$ unabhängige Systemaktionen, so bezeichnen diese atomare Teilprozesse, die in unterschiedlichen *Komponenten* eines verteilten Systems ausgeführt werden, also (in erster Näherung) auf verschiedenen Prozessoren oder Computern (von „künstlicher“ Nebenläufigkeit wie *Multi-Tasking* wollen wir abstrahieren). Allerdings sind solche physikalischen Beispiele nur unzureichend als Erklärung dieses Begriffs: Unter einer Komponente wollen wir allgemein eine *Umgebung* für einen sequentiellen Teilprozeß verstehen. Ist $c$ etwa eine Aktion, die in Kausalitätsbeziehung zu $a$ steht, so ist $c$ in derselben Komponente des Gesamtsystems beheimatet wie $a$, bildet also zusammen mit $a$ und gegebenenfalls weiteren Aktionen einen sequentiellen Prozeß $P_a$, der unabhängig zu dem sequentiellen Prozeß $P_b$ verläuft, zu dem $b$ gehört. Verschiedene, in derselben Komponente eines verteilten Systems beheimatete Aktionen sind also voneinander abhängig und können nicht nebenläufig zueinander ausgeführt werden. Komponenten können sich überlappen (d. h. eine Systemaktion kann in mehr als einer Komponente ausführbar sein) und damit die Interaktion von verschiedenen sequentiellen Prozessen ermöglichen: Ist etwa $c$ auch abhängig von $b$, so werden die Prozesse $P_a$ und $P_b$ durch die gemeinsame Aktion $c$ *synchronisiert*.

Wir weisen in dieser Arbeit nach, daß eine Verteilung von Aktionen eines Spursystems auf Systemkomponenten immer möglich ist, eine maximale Verteilung kann aus der Struktur des Systems rekonstruiert werden kann. Die Existenz von Systemkomponenten kann also auch ohne die explizite Verwendung eines Kompositionsoperators vorausgesetzt werden.

**Systemeigenschaften.** Man ist heute allgemein der Ansicht, daß temporale Logiken adäquate Formalismen zur Spezifikation sog. *reaktiver Systeme* sind, also solcher Systeme, die eine fortwährende Interaktion mit ihrer Umgebung unterhalten. Mithin sind verteilte System i. d. R. reaktive Systeme. Temporale Logiken (zumindest soweit sie im Kontext reaktiver Systeme verwendet werden) sind Erweiterungen der klassischen Aussagenlogik um *Temporaloperatoren*, die Aussagen über das Verhalten eines Systems im Laufe der Zeit erlauben; dabei ist unter „Zeit“ die Abfolge von Systemereignissen zu verstehen. Man unterscheidet *lineare* und *verzweigte* Zeitmodelle. In einem linearen Zeitmodell wird die Gültigkeit einer Formel

für eine mögliche (maximale) Entwicklung eines Systems bestimmt. Diese Formel gilt dann für ein System, wenn sie für jede mögliche Verhaltensweise dieses Systems gilt. Verzweigte Zeitmodelle erlauben die Unterscheidung, ob eine Formel ausgehend von einem bestimmten Systemzustand für wenigstens eine oder aber für alle möglichen Verhaltensweisen eines Systems gelten soll. *Linear time temporal logic* (LTL) und *computation tree logic* (CTL) sind die jeweiligen klassischen Vertreter temporaler Logiken für lineare und verzweigte Zeitmodelle. Die Standardreferenzen zu klassischen (d. h. für eine sequentielle Semantik definierten) temporalen Logiken sind [**28**] und [**82**]. Zu halbordnungsbasierten temporalen Logiken sei der Leser auf [**65, 85**] verwiesen.

Wir stellen in dieser Arbeit die temporale Logik DCTL vor. DCTL operiert über einem verzweigten Zeitmodell, wobei Aussagen möglich sind, die sich auf die Komponentenverteilung und die Kausalstruktur eines verteilten Systems beziehen. Die Semantik von DCTL ist durch die Logik TrPTL [**85**] inspiriert, die über einem linearen Zeitmodell definiert ist (allerdings kann DCTL nicht als die *branching time version* von TrPTL angesehen werden).

Die Verwendung von DCTL anstelle anderer halbordnungsbasierter Logiken ist dadurch motiviert, daß DCTL eine einfache, überschaubare Semantik (ohne versteckte temporale Abhängigkeiten, wie sie in TrPTL auftreten können; vgl. hierzu Bemerkung IV.1.11) hat, die direkt über Semisprachen (d. h. Mengen von Semiordnungen) von Spursystemen definiert werden kann. Daraus resultiert unter anderem ein einfacher Validierungsalgorithmus für DCTL-Formeln, der auf Prozeßautomaten (s. u.) als unterliegende Datenstruktur zur Verhaltensrepräsentation formuliert werden kann.

**Analyse nebenläufiger Systeme.** Wenn ein Spursystem nur endlich viele Zustände einnehmen kann, wird seine Analyse von einem theoretischen Standpunkt aus trivial, da jede seiner Eigenschaften anhand des explizit im Rechner repräsentierten Systems automatisch überprüfbar ist. Allerdings können die zur Analyse einer bestimmten Eigenschaftsklasse benötigten Algorithmen durchaus kompliziert und schwer implementierbar sein, viel gravierender ist jedoch das *Zustandsexplosionsproblem*. Ein Spursystem wird i. Allg. in einer spezifischen Syntax beschrieben. Unter dem *Zustandsexplosionsproblem* verstehen wir das Phänomen, daß die Größe eines Spursystems exponentiell in der Größe seiner Beschreibung sein kann. Bevor wir den Ansatz dieser Arbeit zur (teilweisen) Lösung dieses Problems beschreiben, geben wir einen komprimierten Überblick über Methoden zur Vermeidung bzw. zur Kontrolle dieses exponentiellen Wachstums und über die Analysetechniken, die in Verbindung mit solchen Methoden anwendbar sind. Eine ausführliche Diskussion findet sich in [**91**].

*Binäre Entscheidungsdiagramme (BDDs).* Obwohl diese Arbeit einen halbordnungsbasierten Ansatz verfolgt, sollen BDDs wegen ihres außerordentlichen Erfolgs an dieser Stelle genannt werden. BDDs [**13, 14**] (s. [**63**] für eine Referenz im Zusammenhang mit Petri-Netzen) sind „nichts weiter“ als Datenstrukturen zur

speicherplatzeffizienten Repräsentation von Mengen von Bitvektoren fester Länge. Die Zustände eines nebenläufigen Systems können häufig als solche Bitvektoren repräsentiert werden. Das klassiche Modelchecking-Verfahren für CTL ist auf dieser Datenstruktur einfach formulierbar [**15, 52**]. Neuere Ansätze befassen sich mit Modelchecking für LTL [**18, 77, 96**].

BDDs erlauben das Abspeichern extrem großer Zustandsmengen. Dem Autor ist jedoch nicht bekannt, daß es gelungen wäre, halbordnungsbasierte Verhaltensrepräsentationen mit Hilfe von BDDs effizient zu repräsentieren. Daraus ist die Faustformel abzuleiten: Wenn wir lediglich an zustandsbasierten Aussagen interessiert sind (insbesondere solchen, die in CTL formulierbar sind), sollte ein BDD-basiertes Verfahren die erste Wahl sein. Aussagen über Nebenläufigkeiten und Kausalbeziehungen jedoch erfordern offenbar eine der folgenden Alternativen.

*Spurbasierte Verfahren.* Die den Mazurkiewiczspuren zugrundeliegende Idee, daß unabhängige Symbole in beliebiger Reihenfolge als Systemereignisse auftreten können, wird hier zur Konstruktion eines Transitionssystems verwendet, das möglichst wenige Repräsentanten von Äquivalenzklassen solcher Sequenzen bestimmt, die die Sprache eines gegebenen Spursystems ausmachen. Beispiele sind Godefroids *partial-order methods* [**33**] sowie Valmaris *sture Reduktion* [**88, 89, 90, 92**]. Dabei ist zu bemerken, daß die sture Reduktion nicht nur auf Spursysteme anwendbar ist, sondern auch auf Transitionssysteme mit einer zustandsabhängigen Unabhängigkeitsrelation (vgl. die Diskussion in Abschnitt I.3). Wir ziehen dennoch den Begriff „spurbasierte Verfahren“ dem vielfach verwendeten, jedoch irreführenden Begriff „halbordnungsbasierte Verfahren“ vor. Spurbasierte Verfahren können gut mit LTL Modelchecking-Verfahren kombiniert werden (s. [**65**] für eine Zusammenfassung).

*Halbordnungsrepräsentationen — Auffaltungen.* Die bisher diskutierten Verfahren zielen darauf ab, die Zustandsmenge eines nebenläufigen Systems entweder komprimiert oder aber nur zu einem kleinen Teil darzustellen. Halbordnungsrepräsentationen verzichten vollständig auf die explizite Repräsentation globaler Systemzustände. Die in [**29, 59**] eingeführten *verzweigten Prozesse* bzw. *verzweigten Abläufe* beschreiben das *verzweigte* Systemverhalten sog. sicherer Petri-Netze als (u. U.unendliches) Petri-Netz.[3]

In [**53**] wird ein Verfahren beschrieben, das ein endliches Anfangsstück des — wie sich zeigt — eindeutig bestimmten maximalen verzweigten Ablaufs eines sicheren Petri-Netzes bestimmt, das dennoch genügend Information enthält, um jeden möglichen Ablauf dieses Petri-Netzes zu rekonstruieren. Dieses Anfangsstück wird kurz als *Auffaltung* bezeichnet. Esparza [**30**] formuliert ein Modelchecking-Verfahren für eine eingeschränkte CTL-Variante, das auf einer solchen Auffaltung operiert.

[3]Verzweigte Abläufe sind das Gegenstück zu den abstrakteren Ereignisstrukturen; ein Äquivalent zu der in Ereignisstrukturen explizit gegebene Konfliktrelation kann aus der Graphstruktur eines verzweigten Ablaufs rekonstruiert werden.

In [**31**] wird McMillans Verfahren aus [**53**] verallgemeinert: Der Algorithmus zur Konstruktion der Auffaltung ist bzgl. eines bestimmten Terminationskriteriums formuliert; eine ungünstige Wahl dieses Kriteriums kann dazu führen, daß die Auffaltung eines Petri-Netzes größer als das durch dieses Petri-Netz bestimmte Transitionssystem. Esparza, Römer und Vogler präsentieren nicht nur ein optimales Terminationskriterium, sie charakterisieren auch die Klasse der möglichen in Frage kommenden Kriterien.

Wir verwenden in dieser Arbeit eine teilweise Halbordnungsrepräsentation des Verhaltens eines Spursystems, die wir als *Prozeßautomaten*[4] bezeichnen; allerdings involviert diese Repräsentation auch globale Systemzustände. Die Idee ist: An einigen Punkten (Zuständen) verzweigt sich das Verhalten eines nebenläufigen Systems aufgrund einer nichtdeterministischen Auswahl, an anderen Punkten laufen alternative oder rekurrente Verhaltensweisen zusammen und können gemeinsam fortgesetzt werden. Zwischen diesen Verzweigungs- und Vereinigungspunkten finden durch Semiwörter repräsentierte Abläufe statt, die nebenläufiges Verhalten involvieren können. Gegenüber spurbasierten Verfahren besteht hierbei der Vorteil, daß Nebenläufigkeiten und Kausalstrukturen explizit repräsentiert werden, was bei der Formulierung eines Modelchecking-Verfahrens für eine verteilte Logik nützlich ist.

Die folgenden Vorteile bestehen gegenüber der Verwendung von Auffaltungen: Auffaltungen stellen (da ganz auf globale Zustände verzichtet wird) eine sehr kompakte Verhaltensrepräsentation dar. Das führt aber auch dazu, daß Informationen, die bei anderen Repräsentationen schon während der Erzeugung gewonnen werden können, später u. U. aufwendig aus der Auffaltung extrahiert werden müssen. Die Kreisstruktur eines Spursystems, die sich in der Kreisstruktur eines zugehörigen Prozeßautomaten wiederfindet, ist ein Beispiel. Weitere Beispiele sind Informationen über Systemverklemmungen und Lebendigkeit (vgl. [**23**]).

Ein weiterer Vorteil ist technischer Natur: Vereinigungspunkte sind nicht notwendigerweise lokal. Unter einer lokalen Systementwicklung (die etwa zu einem Vereinigungspunkt führt) wollen wir hierbei eine zu einem einzigen sequentiellen Prozeß gehörende Entwicklung verstehen, wobei nebenläufiges Verhalten in anderen Teilen des Gesamtsystems nicht betrachtet wird (wohl aber Fortschritte anderer interagierender Prozesse, die zum Zustandekommen dieser Systementwicklung notwendig sind). Das bedingt, daß Vereinigungen alternativen oder rekurrenten Systemverhaltens in Prozeßautomaten früher erfolgen kann als in Auffaltungen, da das oben angesprochene Terminationskriterium für die Konstruktion einer Auffaltung lokaler Natur ist (vgl. hierzu auch Bemerkung III.1.14).

Ein weiterer Vorteil ist, daß Prozeßautomaten für allgemeine Spursysteme, Auffaltungen hingegen für Petri-Netze definiert sind (obwohl allerdings Korollar II.2.8 nahelegt, daß jede Syntax zur Beschreibung von Spursystemen zumindest soweit als Petri-Netz aufgefaßt werden kann, um eine Auffaltung daraus zu konstruieren).

[4] Die Namensgebung erfolgte in Anlehnung an den Titel der Arbeit [**78**].

Allerdings erfordert die effiziente Konstruktion von Prozeßautomaten mit Hilfe der in dieser Arbeit vorgestellten Algorithmen eine aufwendige Voranalyse des zugrundeliegenden Spursystems. Darüberhinaus erzeugen diese Algorithmen nicht notwendigerweise in ihrer Größe minimale Prozeßautomaten. Unser Ansatz soll deshalb als komplementäre Technik zu den spurbasierten Verfahren und der Auffaltung verstanden.

**Ähnliche Arbeiten.** Prozeßautomaten wurden von Ulrich [**86, 87**] als *behaviour machines* zum Zweck der Testfallerzeugung eingeführt. Ulrich beruft sich dabei auf Ideen von Probst und Li [**69, 70**]. Ulrich präsentiert einen Konstruktionsalgorithmus, der als Eingabe die Auffaltung eines Petri-Netzes erwartet. In dieser Arbeit wird Ulrichs Ansatz wie folgt verbessert: Zunächst wird der in [**87**] weitgehend informell beschriebene Begriff der *behaviour machine* theoretisch fundiert. Dies erlaubt es, nicht nur die Menge der durch einen Prozeßautomaten akzeptierten Semiwörter genau zu bestimmen, sondern führt auch zu einem Minimalitätskriterium für Prozeßautomaten. Weiterhin kann auf der Basis der präsentierten Theorie ein effizienterer Erzeugungsalgorithmus präsentiert werden, der direkt auf der syntaktischen Repräsentation eines Spursystems operieren kann. Ein Modelchecking-Verfahren, das ebenfalls in dieser Arbeit beschrieben wird, liegt außerhalb von Ulrichs Themenstellung.

Ulrichs Algorithmus weist noch einige weitere Schwächen auf, die unser Algorithmus vermeidet. Zunächst liefert Ulrichs Algorithmus für Systeme, die eine Verklemmungssituation aufweisen, inkorrekte Ergebnisse; Ulrich setzt deshalb auf eine Voranalyse zum Ausschluß verklemmungsbehafteter Systeme. Weiterhin muß die Auffaltung, die Eingabe für Ulrichs Algorithmus ist, unter Verwendung eines bestimmten Terminations-Kriteriums erzeugt worden sein: Unter allen möglichen Kriterien wird das ineffizienteste benötigt.

Vernadat et. al. [**93, 94**] verfolgen einen zum Prozeßautomaten ähnlichen Ansatz, der als *step covering graph* bezeichnet wird. Diese Graphen sind — grob gesprochen — Prozeßautomaten, die nur solche Semiwörter mit leerer Ordnungsrelation enthalten. Der Erzeugungsalgorithmus für *step covering graphs* ähnelt dem in dieser Arbeit vorgestellten Grundalgorithmus III.1 (vgl. Bemerkung III.2.5).

Hulgaard et. al. [**42, 43, 44**] schließlich verwenden eine ebenfalls als *process automaton* bezeichnete Struktur zur Verifikation und Leistungsanalyse zeitbehafteter nebenläufiger Systeme. *Process automata* und Prozeßautomaten unterscheiden sich lediglich dadurch, daß die erstere Struktur Abläufe eines sicheren Petri-Netzes zur Repräsentation nebenläufigen Verhaltens anstelle von Semiordnungen verwendet — wir übersetzen deshalb *process automaton* mit *Ablaufautomat*. In [**42**] wird ein Erzeugungsalgorithmus präsentiert, der als Eingabe ein stur reduziertes Transitionssystem zu einem sicheren Petri-Netz erwartet. Zwar bestehen gewisse Analogien zwischen den in dieser Arbeit präsentierten Algorithmen zur Konstruktion von Prozeßautomaten und der sturen Reduktion, allerdings ist unklar, wie sich beide Ansätze in Beziehung setzen lassen.

Eine weitere Arbeit, die einen ähnlichen Ansatz wie diese Arbeit beschreibt, ist [**40**]. Hier wird eine Modelchecking-Algorithmus für eine temporale Fixpunktlogik beschrieben, der auf der Auffaltung eines sicheren Petri-Netzes beruht. Inwieweit sich dieser Algorithmus in der Praxis besser oder schlechter als der in dieser Arbeit präsentierte Modelchecker verhält, muß durch Experimente bestimmt werden. Wir vermuten jedoch, daß die bereits diskutierte Notwendigkeit, Informationen nachträglich (also während des Modelchecking-Prozesses) zu erlangen, dem in [**40**] beschriebenen Verfahren einen gewissen Nachteil bringen wird.

**Organisation der Arbeit.** Die Arbeit ist in vier Kapitel unterteilt, jedes dieser Kapitel umfaßt drei Abschnitte, die wiederum in eine variable Anzahl nicht nummerierter Unterabschnitte unterteilt sind. Definitionen, Theoreme, Lemmata, Beispiele usw. sind insgesamt fortlaufend in der Form $K.i.j$ nummeriert; $K$ ist dabei die Nummer des jeweiligen Kapitels, $i$ die Nummer des Abschnitts und $j$ die Nummer des Paragraphen, in dem das jeweilige Theorem, Lemma, die Definition oder das Beispiel usw. zu finden ist.

Inhaltlich ist die Arbeit wie folgt organisiert:

*Kapitel I: Grundlagen.* Dieses Kapitel erklärt die mathematischen Grundlagen der Arbeit. Das hier präsentierte Material besteht größtenteils in einer Aufbereitung wohlbekannter Erkenntnisse über beschriftete Halbordungen, Spursysteme und Mazurkiewiczspuren. Abschnitt I.1 enthält grundlegende Definitionen und Schreibweisen. Mengentheoretische und prädikatenlogische Schreibweisen sowie der Begriff der Funktion sind als bekannt vorausgesetzt, jeder darüber hinausgehender Begriff wird jedoch eingeführt, da wir einige Schreibweisen verwenden, die von den sonst üblichen abweichen.

Abschnitt I.2 behandelt beschriftete Halbordnungen, Semiordnungen, Halbwörter und Semiwörter. Der Präfixbegriff und der Begriff der Sequentialisierung werden eingeführt; wir weisen nach, daß für Semiordnungen sowohl die Präfixbildung als auch die Sequentialisierung eindeutig geschehen kann. Wir übernehmen einige der in [**78**] definierten Operationen auf Halb- bzw. Semiwörtern und definieren weitere, in dieser Arbeit nützliche Operationen. Kanonische Darstellungen von Semiordnungen werden behandelt; dabei ist eine Darstellung kanonisch, wenn isomorphe Semiordnungen gleich dargestellt werden. Eine solche kanonische Darstellung erlaubt es uns, Durchschnitt und Vereinigung einer Menge von Semiordnungen als *meet* bzw. *join* im Sinne der Verbandstheorie [**10**] zu definieren.

Abschnitt I.3 diskutiert Transitionsysteme und Spursysteme. Wir verwenden Petri-Netze (genauer: eine Variante elementarer Netzsysteme) als syntaktische Repräsentation für Spursysteme, allerdings sei darauf hingewiesen, daß es Spursysteme gibt, die nicht mit Hilfe eines Netzsystems beschreibbar sind; unsere Überlegungen beziehen sich aber tatsächlich auf Spursysteme, Petri-Netze werden nur in Beispielen und als Motivationshilfen verwendet. Wir ordnen Spursystemen zwei Semantiken zu, nämlich Mazurkiewiczspuren und Semiwörter; die klassische sequentielle Semantik ergibt sich dann als ein Spezialfall der Semiwortsemantik.

Wir stellen uns die Frage, inwieweit eine Semiordnung die tatsächlichen kausalen Abhängigkeiten und Nebenläufigkeiten in einem Spursystem beschreibt: Wir bestimmen konstruktiv eine Klasse von Semiordnungen, in der die Begriffe der Anordnung und Beziehungslosigkeit und denen der kausalen Abhängigkeit und Unabhängigkeit übereinstimmen. Abschließend weisen wir nach, daß beide Semantiken äquivalent im Sinne der allgemeinen Algebra sind: Das Spurmonoid und das Semiwortmonoid bzw. das kanonische Semiordnungsmonoid sind isomorph.

*Kapitel II: Komponenten und verteilte Systeme.* Wir befassen uns nun mit der Frage, inwieweit ein nebenläufiges System (repräsentiert durch ein Spursystem) als verteiltes System aufgefaßt werden kann. Zu diesem Zweck definieren wir in Abschnitt II.1 sog. Positionszuweisungen an Spursysteme. Die Idee ist hierbei, daß sich Komponenten in verteilten Systemen wenigstens durch die Existenz von zu ihnen lokalen Systemzuständen auszeichnen. Positionszuweisungen an Systemaktionen beschreiben, welche Komponenten durch das Stattfinden einer Systemaktion beeinflußt werden. Positionszuweisungen an globale Systemzustände beschreiben, in welchen lokalen Zuständen sich die involvierten Komponenten befinden. Dabei ist aber nicht vorausgesetzt, daß globale Zustände bereits vollständig durch eine Menge lokaler Zustände charakterisiert sind.

Abschitt II.2 enthält das folgende Ergebnis: Jedes Spursystem kann in nichttrivialer Weise als verteiltes System aufgefaßt werden. Wir erziehlen dieses Ergebnis durch eine Anwendung eines Theorems aus der Theorie der Regionen [**62**] (andere Referenzen zu diesem Thema sind [**3, 2, 6, 21, 27**]). Zwar ist die Methode, mit der eine solche Verteilung bestimmt werden kann, konstruktiv, erfordert jedoch die explizite Konstruktion des zugrundeliegenden Spursystems. Wir gehen deshalb nicht davon aus, daß die im folgenden verwendeten Positionszuweisungen die Kriterien für eine nichttriviale Positionszuweisung erfüllen.

Abschnitt II.3 definiert auf der Grundlage der in den beiden vorhergehenden Abschnitten angestellten Überlegungen den Begriff des *verteilten Systems*. Wir geben drei Beispiele für Ausprägungsformen für verteilte Systeme an: Platzinvariantenüberdeckte kontaktfreie Netzsysteme, synchron und asynchron kommunizierende Zustandsmaschinen.

*Kapitel III: Prozeßautomaten.* Dieses Kapitel befaßt sich mit der Definition, den grundlegenden Eigenschaften und der Erzeugung von Prozeßautomaten. Abschnitt III.1 definiert Prozeßautomaten und diskutiert die Begriffe der Verzweigungs-, Vereinigungs- und Rekurrenzvollständigkeit. Abschnitt III.2 stellt einen ersten Erzeugungsalgorithmus für Prozeßautomaten für ein gegebenes Spursystem vor. Dabei ist es nicht notwendig, dieses Spursystem explizit aufzubauen; der Algorithmus operiert auf der Syntax, in der das Spursystem beschrieben wird. Wir beweisen die totale Korrektheit dieses Grundalgorithmus.

Abschnitt III.3 diskutiert verschiedene Möglichkeiten zur Verbesserung des Grundalgorithmus: Die Entfernung redundanter Zustände, das Verschieben der

Betrachtung konzessionierter Aktionen, und schließlich die Verwendung der Komponentenstruktur, die wir aufgrund der in Kapitel II angestellten Überlegungen für ein Spursystem voraussetzen dürfen. In diesem Zusammenhang wird auch ein Begriff aus der Petri-Netz-Theorie verwendet, nämlich der *Falle*.

*Kapitel IV: Eine Logik für verteilte Systeme.* Wir untersuchen eine verteilte, agentenbasierte Version der klassischen temporalen Logik CTL, die wir als *distributed computation tree logic (DCTL)* bezeichnen. Abschnitt IV.1 beschreibt die Syntax und die Semantik dieser Logik und diskutiert verschiedene Möglichkeiten, abgeleitete temporale (und aussagenlogische) Operatoren zu definieren. Weiterhin wird das Verhältnis von DCTL zu CTL bestimmt. Wir reden auch über eine einfache Klasse globaler Eigenschaften, die in DCTL nicht oder nur schwer formulierbar sind. Wir zeigen anhand einiger Beispiele auf, daß globale Eigenschaften vielfach als lokale Eigenschaften beschreibbar sind. Weiterhin wird demonstriert, wie durch eine Modifikation eines verteilten Systems (d. h. durch Hinzufügen eines *Beobachters* einer globalen Eigenschaft) globale Eigenschaften formuliert werden können, falls eine lokale Beschreibung dieser Eigenschaft nicht gelingt oder aber zu aufwendig ist.

Um zu demonstrieren, wie Systemeigenschaften mit Hilfe von DCTL formuliert werden können, behandelt Abschnitt IV.2 ein größeres Anwendungsbeispiel. Es werden Aspekte eines verteilten Datenbanksystems spezifiziert. Es zeigt sich, daß Eigenschaften eines solchen Systems elegant in DCTL formuliert werden können. Wir zeigen ausserdem auf, daß einige dieser Eigenschaften nicht in CTL formulierbar sind.

Abschnitt IV.3 stellt einen Algorithmus zur automatischen Überprüfung der Frage vor, ob eine gegebene DCTL-Formel in einem gegebenen verteilten System gilt oder nicht; solche Algorithmen werden als *Modelchecker* bezeichnet. Die beschriebene Technik ist eine Variante der klassischen Modelchecking-Techniken für Logiken über einem verzweigten Zeitmodell.

*Ausblick.* Ein Ausblick, in dem auch Details zu einer möglichen Implementierung der in dieser Arbeit beschriebenen Algorithmen erläutert werden, schließt die Arbeit ab.

Abb. 1 stellt den Zusammenhang der Inhalte der Arbeit noch einmal graphisch dar. Ausgehend vom Begriff des Spursystems wird einerseits eine Halbordnungssemantik definiert, andererseits der Begriff des verteilten Systems eingeführt; beide Themen sind weitgehend unabhängig voneinander. Die Halbordnungssemantik ist Voraussetzung für die Definition des Prozeßautomaten und für die erste Formulierung eines Generierungsalgorithmus. Verteilte Systeme werden erst benötigt, wenn Optimierungen des Grundalgorithmus diskutiert werden. Die Definition von DCTL ist wiederum unabhängig von der des Prozeßautomaten, verwendet jedoch sowohl die Halbordnungssemantik für Spursysteme als auch den Begriff des verteilten Systems. Zur Formulierung des *Modelcheckers* schließlich wird DCTL wie auch der Prozeßautomat benötigt.

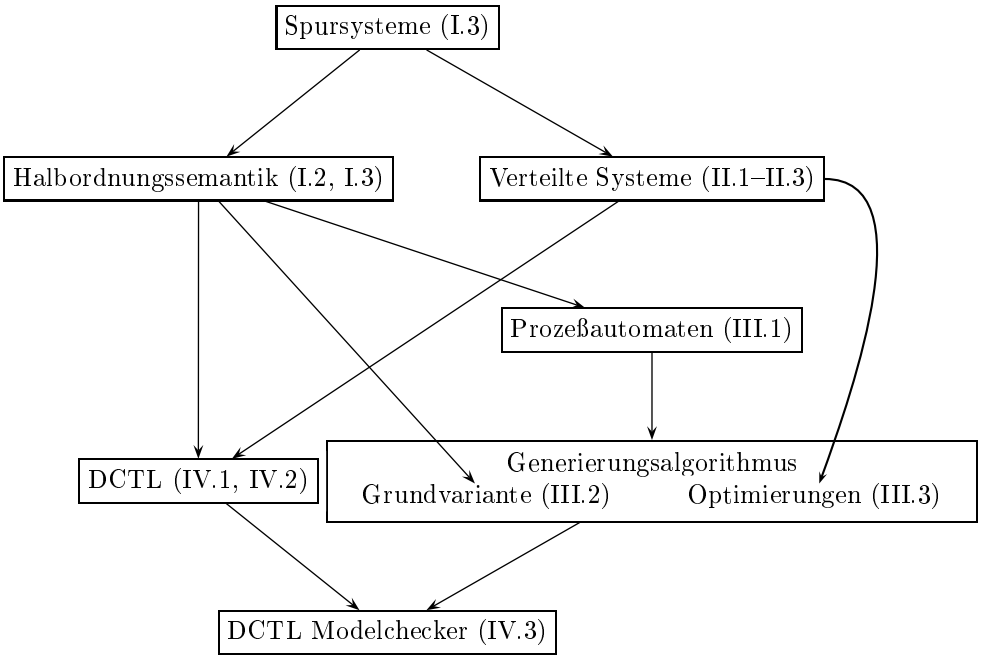



ABBILDUNG 1. Struktur der Arbeit

**Technische Anmerkungen.** Diese Arbeit wurde mit dem außerordentlichen Satzsystem TEX [**47**] von Donald Knuth unter Verwendung von Lesley Lamports Makrosatz LATEX [**34**] gesetzt; weiterhin wurden verschiedene weitere Makrosätze verwendet, insbesondere die Makropackete der American Mathematical Society. Graphiken der initialen Version dieser Arbeit wurden mit `xfig` [**97**] erstellt, weiterhin kam der Petri-Netz-Editor PED [**64**] zur Anwendung. Die Graphiken in dieser überarbeiteten Fassung wurden mit Hilfe des TEX-Markopackets pstricks [**98**] „programmiert“. Sogar Knuths Fonterzeugungssystem METAFONT [**46**] wurde benutzt. Eine prototypische Implementierung der Algorithmen, die in Kapitel III beschrieben sind, wurde in der Programmiersprache ADA [**4, 83**] geschrieben; der verwendete Compiler war `gnat` (GNU ADA Translator).

KAPITEL I

# Grundlagen

Dieses Kapitel erklärt die mathematischen Grundlagen dieser Arbeit. Schreibweisen und grundlegende Definitionen werden in Abschnitt I.1 eingeführt.

In Abschnitt I.2 definieren wir die Begriffe der Semiordnung und des Semiworts als mathematische Strukturen zur Erklärung der Semantik sog. Spursysteme, die als allgemeines Systemmodell Verwendung finden. Kanonische Semiordnungen werden eingeführt, die zu den Begriffen des Durchschnitts als größte untere Schranke und der Vereinigung als kleinste obere Schranke einer Menge von Semiwörten im Sinne der Verbandstheorie führen.

Abschnitt I.3 beschreibt Spursysteme, denen wir zwei Semantiken zuordnen, Mazurkiewiczspuren und Semiwörter. Semiwörter kleinster Sequentalität werden eingeführt. Wir zeigen die Äquivalenz beider Semantiken.

## I.1. Mathematische Grundbegriffe

Mit Ausnahmen bezeichnen wir Strukturen mit kalligraphischen Großbuchstaben $\mathscr{A}, \mathscr{B}, \ldots$, während Familien $\{A_i\}_{i \in I}$ für eine Indexmenge $I$ und Mengen $A_i$ $(i \in I)$ durch schräggestellte Großbuchstaben $\boldsymbol{A}$, $\boldsymbol{B}$, ... im Fettdruck notiert werden. Für spezielle Mengen oder Sprachen verwenden wir aufrechte fettgedruckte Buchstaben oder Buchstabenfolgen. So ist etwa $\mathbf{L}(\mathscr{T})$ die Sprache eines Transitionssystems $\mathscr{T}$ und $\mathbf{TR}(\Sigma)$ die Menge der Spuren über $\Sigma$ (Abschnitt I.3). Schließlich wählen wir aufrechte Buchstaben und Buchstabenfolgen als Benennungen spezieller Operationen (min, lin, tr). Kleine schräggestellte fettgedruckte Buchstaben $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ stehen für Halb- bzw. Semiwörter (Abschnitt I.2).

Zur Reduktion des Schreibaufwands verwenden wir die folgende Konvention: Wird eine Struktur $\mathscr{S} = \langle A, B, \ldots \rangle$ eingeführt, so werden ihre Komponenten mit $A_{\mathscr{S}}, B_{\mathscr{S}}, \ldots$ bezeichnet. Ist aber $B_{\mathscr{S}} = \langle C, D, \ldots \rangle$ wiederum eine Struktur, so schreiben wir $C_{\mathscr{S}}, D_{\mathscr{S}}, \ldots$ anstelle von $C_{B_{\mathscr{S}}}, D_{B_{\mathscr{S}}}, \ldots$, wenn eine Verwechslung ausgeschlossen ist, um „Indextreppen“ zu vermeiden.

Mit $\mathbb{N}$ bezeichnen wir die Menge der natürlichen Zahlen (einschließlich 0), mit $\mathbb{Z}$ die der ganzen Zahlen. Boolsche Werte sind $\mathbb{B} = \{\mathrm{false}, \mathrm{true}\}$. Ist $n \in \mathbb{N}$, so ist

$$[n] =_{\mathrm{Def}} \{m \in \mathbb{N} : m < n\}.$$

$\mathscr{P}(A)$ steht für die *Potenzmemge* von einer Menge $A$, d. h. die Menge aller Teilmengen von $A$. $\mathscr{P}_{\mathrm{f}}(A)$ bezeichnet die Menge aller endlichen Teilmengen von $A$.

**Relationen und Abbildungen.** Für Relationen $R \subseteq A \times B$ verwenden wir häufig die Infixschreibweise, d. h.

$$a\ R\ b \Leftrightarrow_{\mathrm{Def}} \langle a, b\rangle \in R.$$

Ist $R \subseteq A \times B$ eine Relation und ist $a \in A$, so bezeichnet

$$R(a) =_{\mathrm{Def}} \{b \in B : a\ R\ b\}$$

das *Bild* von $a$ unter $R$. Diese Notation wird durch

$$R(C) =_{\mathrm{Def}} \bigcup_{a \in C} R(a)$$

auf Teilmengen $C \subseteq A$ erweitert. $R^{-1} \subseteq B \times A$ steht für die *inverse Relation* zu $R$, d. h.

$$b\ R^{-1}\ a \Leftrightarrow_{\mathrm{Def}} a\ R\ b.$$

Für jede Menge $A$ bezeichnet $\mathrm{id}_A \subseteq A \times A$ die *Identitätsrelation* auf $A$, d. h.

$$a\ \mathrm{id}_A\ b \Leftrightarrow_{\mathrm{Def}} a = b.$$

Ist $R \subseteq A \times A$ eine Relation, so bezeichnen wir mit $\bar{R}$ ihr *Komplement*, d. h. die Relation $(A \times A) - R$. Dabei wird sich jeweils aus dem Zusammenhang ergeben, welche Menge $A$ gemeint ist.

Die *Komposition* $R \cdot S \subseteq A \times C$ zweier Relation $R \subseteq A \times B$ und $S \subseteq B \times C$ ist als

$$a\ (R \cdot S)\ c \Leftrightarrow_{\mathrm{Def}} \exists b \in B\,(a\ R\ b\ \&\ b\ S\ c)$$

definiert.

Eine Menge $B \subseteq A$ heißt *abgeschlossen* bzgl. einer Relation $R \subseteq A \times A$, wenn $R(B) \subseteq B$ gilt.

Zuweilen machen wir Gebrauch von partiellen Funktionen $f$ mit *Argumentbereich* $A$ und *Wertebereich* $B$ und verwenden hierfür die Notation $f : A \rightharpoonup B$, während totale Funktionen wie gewohnt als $f : A \to B$ notiert werden. Wir unterscheiden nicht zwischen Abbildungen $f : A \rightharpoonup B$ und ihren *Graphen*

$$\{\langle a, b\rangle \in A \times B : f(a) \text{ definiert und } b = f(a)\}.$$

Dies erlaubt es uns, mengentheoretische Operationen wie $\cup$ oder $-$ auf Abbildungen anzuwenden. Wir schreiben aber wie gewohnt $b = f(a)$ anstelle von $b \in f(a)$. Die *Komposition* zweier Abbildungen $f : A \rightharpoonup B$ und $g : B \rightharpoonup C$ wird wie gewohnt als

$$(f \circ g)(b) =_{\mathrm{Def}} f(g(b))$$

definiert; für Funktionen gilt also $\{(f \circ g)(a)\} = (g \cdot f)(a)$. Wir bezeichnen die Menge der partiellen Abbildungen von $A$ nach $B$ mit $(A \rightharpoonup B)$ und die der totalen Abbildungen von $A$ nach $B$ mit $(A \to B)$. Die *Restriktion* $f \restriction C$ einer Abbildung $f : A \rightharpoonup B$ auf eine Teilmenge $C \subseteq A$ ist als

$$f \restriction C =_{\text{Def}} f \cap (C \times B)$$

definiert.

Eine Relation $R \subseteq A \times A$ ist

(a) *reflexiv*, wenn $\text{id}_A \subseteq R$ gilt;
(b) *irreflexiv*, wenn $\text{id}_A \cap R = \varnothing$ gilt;
(c) *symmetrisch*, wenn $R^{-1} \subseteq R$ gilt;
(d) *antisymmetisch*, wenn $R^{-1} \cap R = \varnothing$ gilt;
(e) *transitiv*, wenn $R \cdot R \subseteq R$ gilt.

Sei $R \subseteq A \times A$ eine Relation über $A$. Mit $R^+$ bezeichnen wir die kleinste Relation $S \subseteq A \times A$ mit $R \subseteq S$ und $S \cdot S \subseteq S$, während $R^*$ für $R^+ \cup \text{id}_A$ steht. $R^+$ heißt *transitive Hülle* über $R$, $R^*$ wird *reflexiv-transitive Hülle* über $R$ genannt.

Ist $R \subseteq A \times A$ eine symmetrische Relation, so bezeichnen wir eine Menge $C \subseteq A$ als *$R$-Clique* in $A$ oder als *Clique* bzgl. $R$ in $A$, wenn für alle $a, b \in C$ aus $a \neq b$ bereits $a \; R \; b$ folgt. Ist $C$ eine $R$-Clique in $A$, für die $C \subseteq D \Rightarrow C = D$ für alle weiteren $R$-Cliquen $D$ in $A$ gilt, so heißt $C$ *maximale $R$-Clique* in $A$.

**Äquivalenzen und Kongruenzen.** Eine *Äquivalenzrelation* über einer Menge $A$ ist eine reflexive, symmetrische und transitive Relation $\equiv \subseteq A \times A$. Ist $\equiv$ eine Äquivalenzrelation über $A$, so bezeichnen wir die *Äquivalenzklasse* eines Elements $a \in A$ mit

$$[a]_\equiv =_{\text{Def}} \{b \in A : a \equiv b\},$$

wobei wir uns aber die Freiheit nehmen, den Index $\equiv$ wegzulassen oder im jeweiligen Fall durch ein einfacheres Symbol zu ersetzen. Ist $\mathscr{S} = \langle A, f, g, \ldots \rangle$ eine Struktur mit der Trägermenge $A$ und Operationen $f : A \to A$ oder $g : A \times A \to A$, so heißt eine $\equiv \subseteq A \times A$ *Kongruenz* über $\mathscr{S}$, wenn aus $f(a) \equiv f(b)$ aus $a \equiv b$ bzw. $g(a_1, a_2) \equiv g(b_1, b_2)$ aus $a_1 \equiv b_1$ und $a_2 \equiv b_2$ folgt; für mehrstellige Operationen erfolgt die Begriffsbildung analog.

Spezielle algebraische Strukturen, die wir in Abschnitt I.3 in Verbindung mit Kongruenzen betrachten werden, sind Monoide. Ein Monoid ist eine Struktur $\mathscr{M} = \langle A, \cdot, 1 \rangle$ mit einer Trägermenge $A$, einer Konstanten $1 \in A$ und einer Operation $\cdot : A \times A \to A$, die die Gleichungen

(a) $a \cdot 1 = 1 \cdot a = a$,
(b) $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

für alle $a, b, c \in A$ erfüllt.

**Ordnungen.** Eine *Quasiordnung* über einer Menge $A$ ist eine reflexive und transitive Relation $\leqslant \subseteq A \times A$. Wenn $\leqslant$ ($\preccurlyeq$, $\sqsubseteq$) eine Quasiordnung über $A$ bezeichnet, so steht $<$ für $\leqslant - \mathrm{id}_A$ ($\prec$ für $\preccurlyeq - \mathrm{id}_A$, $\sqsubset$ für $\sqsubseteq - \mathrm{id}_A$). $a \not< b$ bedeutet $\neg(a < b)$ (analog für $\leqslant$, $\prec$, $\preccurlyeq$, $\sqsubset$, $\sqsubseteq$).

Ist $\leqslant \subseteq A \times A$ eine Quasiordnung über $A$, so bezeichnen wir mit

$$\min_{\leqslant}(A) =_{\mathrm{Def}} \{a \in A : \forall b \in A\,(b \not< a)\} \quad \text{und}$$
$$\max_{\leqslant}(A) =_{\mathrm{Def}} \{a \in A : \forall b \in A\,(a \not< b)\}$$

die Mengen der *minimalen* und *maximalen* Elemente von $A$ bzgl. $<$.

Eine *Halbordnung* über $A$ ist eine antisymmetische Quasiordnung über $A$. Eine Halbordnung $\leqslant \subseteq A \times A$ wird *lineare Ordnung* oder *totale Ordnung* genannt, wenn $a = b \vee a < b \vee b < a$ für alle $a, b \in A$ gilt.

Ist $\leqslant \subseteq A \times A$ eine Halbordnung über $A$, so nennen wir ein Element $a \in A$ *obere Schranke* (*untere Schranke*) einer Teilmenge $B \subseteq A$, wenn $b \leqslant a$ ($a \leqslant b$) für alle $b \in B$ gilt. Wenn weiterhin $a \leqslant c$ ($c \leqslant a$) für jede weitere obere Schranke (untere Schranke) $c$ von $B$ vorliegt, wird $a$ als *kleinste obere Schranke* (*größte untere Schranke*) von $B$ bezeichnet.

**Sequenzen.** Eine endliche nicht leere Menge $A$ nennen wir auch ein *Alphabet*. Ist $A$ ein Alphabet, so bezeichnen wir mit $A^*$ die Menge aller endlichen Sequenzen über $A$ inklusive der *leeren Sequenz* $\epsilon$. Ist $\sigma = a_0 a_1 \ldots a_{n-1} \in A^*$, so ist $|\sigma| = n$ die *Länge* von $\sigma$, wobei wir $|\epsilon| =_{\mathrm{Def}} 0$ setzen. Wir fassen Sequenzen $\sigma = a_0 a_1 \ldots a_{n-1}$ auch als Abbildungen $\sigma : i \mapsto a_i$ für $i \in [n]$ auf; damit ist $\epsilon = \varnothing$.[1]

Die *Konkatenation* $\sigma \cdot \rho$ zweier Sequenzen $\sigma, \rho \in \Sigma^*$ ist eine Sequenz der Länge $|\sigma| + |\rho|$, die als

$$(\sigma \cdot \rho)(i) =_{\mathrm{Def}} \begin{cases} \sigma(i), & \text{falls } i < |\sigma|; \\ \rho(i - |\sigma|), & \text{sonst} \end{cases} \qquad (i \in [|\sigma| + |\rho|])$$

definiert ist; wir schreiben auch $\sigma\rho$ anstelle von $\sigma \cdot \rho$. Die Struktur $\mathscr{M}(A) =_{\mathrm{Def}} \langle A^*, \cdot, \epsilon \rangle$ ist das Standardbeispiel für ein Monoid; wir nennen $\mathscr{M}(A)$ das *freie Monoid* über $A$.

## I.2. Semiwörter

Betrachten wir sequentielle Systeme, sind Folgen von Systemaktionen die mögliche Grundlage eines adäquaten Semantikbegriffs. Sind Aktionen wie in verteilten oder parallelen Systemen jedoch teilweise unabhängig voneinander ausführbar, erscheint der Begriff der Halbordnung (anstelle der totalen Ordnung, die dem Sequenzbegriff zugrunde liegt) als mathematische Struktur einer Semantikdefinition jedoch geeigneter. Die Elemente der Trägermenge einer Halbordnung werden als Systemereignisse, die partielle Anordnung dieser Elemente als Kausalordnung

[1] In Abschnitt I.2 werden wir die Definition von Sequenzen nocheinmal auf Strukturen verallgemeinern, die isomorph zu solchen Abbildungen sind.

interpretiert. Ereignisse, die sich nicht in kausalem Zusammenhang befinden, beschreiben nebenläufig ausgeführte Systemaktionen. Wir betrachten endliche Systeme mit einem definierten Startzustand, also solche Systeme mit einer endlichen Anzahl möglicher Zustände und einer endlichen Anzahl von Aktionen, die in einem bestimmten Zustand ausgeführt werden können. In solchen Systemen kann jedes Ereignis nur endlich viele Wirkungen haben. Die Existenz eines Startzustands impliziert, daß jedes Ereignis $e$ eine endliche Vorgeschichte besitzt, also nur endlich viele Ereignisse $e'$ mit $e' < e$ existieren. Da wir jedoch im folgenden nur endliche Halbordnungen betrachten werden, sind beide Annahmen schon erfüllt.

Neben dieser „internen“ Systemsicht können wir auch den Standpunkt eines externen Beobachters einnehmen, der Systemverhalten als Folgen von Systemereignissen wahrnimmt. Die nebenläufige Ausführung der Systemaktionen $a$ und $b$ wird dann so wahrgenommen, daß $a$ in manchen Ausführungsfolgen vor $b$, in anderen hingegen nach $b$ auftritt, je nachdem, welchen Standpunkt der Beobachter gegenüber dem System einnimmt.

Mit diesem Zugang, auf dem der Begriff der Spur aufbaut, werden wir uns in Abschnitt I.3 ausführlich auseinandersetzen; in diesem Abschnitt befassen wir uns zunächst mit Halbordnungen.

**Definitionen und grundlegende Eigenschaften.** Eine beschriftete Halbordnung über einem Alphabet $A$ ist eine Struktur $x = \langle E, <, \lambda\rangle$, wobei $E$ eine endliche Menge von *Ereignissen* ist, $< \subseteq E \times E$ eine als *Kausalordnung* bezeichnete Halbordnung und $\lambda : E \to A$ eine *Beschriftungsfunktion*. Wir bezeichnen die Klasse aller beschrifteten Halbordnungen über $A$ mit $\mathbf{LPO}(A)$.

Sei $x$ eine beschriftete Halbordnung über einem Alphabet $A$. Wir verwenden die folgenden Begriffe und Schreibweisen:

(a) $e_1 \lessdot_x e_2 \Leftrightarrow_{\text{Def}} e_1 <_x e_2 \;\&\; \forall e \in E_x\,(e_1 \leqslant_x e \leqslant_x e_2 \Rightarrow e_1 = e \vee e_2 = e)$.

(b) $e_1 \;\mathrm{co}_x\; e_2 \Leftrightarrow_{\text{Def}} e_1 \not\leqslant_x e_2 \;\&\; e_2 \not\leqslant_x e_1$. Eine Menge $C \subseteq E_x$ wird als *unabhängig* bezeichnet, wenn $e_1 \neq e_2 \Rightarrow e_1 \;\mathrm{co}_x\; e_2$ für alle $e_1, e_2 \in C$ gilt. Eine maximale unabhängige Menge heißt *Schnitt*.

(c) $e_1 \;\mathrm{li}_x\; e_2 \Leftrightarrow_{\text{Def}} e_1 \leqslant_x e_2 \vee e_2 \leqslant_x e_1$, d. h. $\mathrm{li}_x = \bar{\mathrm{co}}_x$. Eine Menge $D \subseteq E_x$ wird als *Kette* bezeichnet, wenn $e_1 \;\mathrm{li}_x\; e_2$ für alle $e_1, e_2 \in D$ gilt.

(d) $\min(x) =_{\text{Def}} \min_{\leqslant_x}(E_x)$ und $\max(x) =_{\text{Def}} \max_{\leqslant_x}(E_x)$.

(e) Ist $a \in A$, so nennen wir die beschriftete Halbordnung $\langle\{a\}, \varnothing, a \mapsto a\rangle$ einen *Buchstaben*. Wenn Verwechslungen ausgeschlossen sind, verwenden wir das Symbol $a$, um sowohl Elemente $a \in A$ wie auch ihre zugehörigen Buchstaben zu bezeichnen.

(f) Die *leere beschriftete Halbordnung* ist $\epsilon = \langle\varnothing, \varnothing, \varnothing\rangle$.

I.2.1. Beispiel. Abbildung I.1 zeigt zwei beschriftete Halbordnungen $x$ und $y$. Wie üblich verwenden wir *Hassediagramme* zur Darstellung von Halbordnungen: Ereignisse werden als Kreise gezeichnet, anstelle der vollständigen Kausalordnung $<$ wird lediglich $\lessdot$ durch Kanten dargestellt. Allerdings nehmen wir uns die Freiheit, zusätzliche transitive

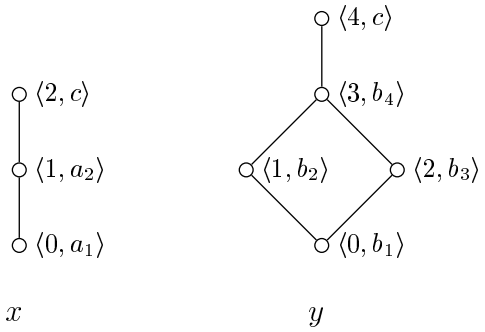



ABBILDUNG I.1. Beschriftete Halbordnungen.

Kanten zwischen geordneten Ereignissen hinzuzufügen, wann immer dies zur Verdeutlichung eines bestimmten Sachverhalts nützlich ist. Gilt $e_1 \lessdot e_2$, so wird das Ereignis $e_2$ über einem Ereignis $e_1$ angeordnet. Ereignisse $e$ werden mit Paaren $\langle e, \lambda(e)\rangle$ beschriftet.

In unserem Beispiel ist also

$$\begin{aligned} x &= \langle \{0,1,2\}, \{\langle 0,1\rangle, \langle 0,2\rangle, \langle 1,2\rangle\}, \{\langle 0,a_1\rangle, \langle 1,a_2\rangle, \langle 2,c\rangle\}\rangle, \\ y &= \langle \{0,1,2,3,4\}, \{\langle 0,1\rangle, \langle 0,2\rangle, \langle 0,3\rangle, \langle 0,4\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 2,3\rangle, \langle 2,4\rangle, \langle 3,4\rangle\}, \\ &\quad \{\langle 0,b_1\rangle, \langle 1,b_2\rangle, \langle 2,b_3\rangle, \langle 3,b_4\rangle, \langle 4,c\rangle\}\rangle \end{aligned}$$

□

I.2.2. BEMERKUNG (Unendliche beschriftete Halbordnung). Sollen unendliche beschriftete Halbordnungen $x$ (d. h. beschriftete Halbordnungen mit unendlicher Ereignismenge) betrachtet werden, wäre zusätzlich zu fordern, daß (1) die Menge $\leqslant_x^{-1}(e)$ für alle $e \in E_x$ endlich ist. Diese Bedingung besagt, daß jedes Ereignis eine endliche Vorgeschichte hat.

Systemzustände sind in beschrifteten Halbordnungen implizit durch bzgl. $\leqslant_x^{-1}$ abgeschlossene endliche Mengen repräsentiert: Eine solche Menge $D$ bezeichnet den Zustand, in dem sich ein System nach dem Stattfinden aller Ereignisse in $D$ befindet. Da zwischen abgeschlossenen Mengen $D$ einer Halbordnung und ihren unabhängigen Mengen $C$ eine Bijektion $D \mapsto \max_{\leqslant_x}(D)$ existiert (die Umkehroperation ist $C \mapsto \leqslant_x^{-1}(C)$), können wir Zustände auch mit unabhängigen Mengen $C$ einer beschrifteten Halbordnung assoziieren. Da wir nur endliche Systeme betrachen wollen, muß jeder Zustand durch das Stattfinden einer endlichen Anzahl von Systemereignissen zustandekommen bzw. beschreibbar sein: Wir fordern deshalb, daß (2) jede unabhängige Menge von $x$ endlich ist.

(1) und (2) implizieren dann, daß $\leqslant_x^{-1}(C)$ für jede unabhängige Menge $C$ von $x$ endlich ist. □

I.2.3. BEMERKUNG (Sequenzen). In Abschnitt I.1 wurde bereits festgestellt, daß Sequenzen $\sigma \in A^*$ als Abbildungen $\sigma : [|\sigma|] \to A$ aufgefaßt werden können. Wir können dann Sequenzen auch als beschriftete Halbordnungen $\langle [|\sigma|], <, \sigma\rangle$ auffassen, wobei $<$ die

Restriktion der gewöhnlichen Kleiner-Ordnung auf $\mathbb{N}$ auf die Menge $[|\sigma|]$ ist. Wir unterscheiden deshalb nicht mehr zwischen Sequenzen $\sigma$ und beschrifteten Halbordnungen der Form $\langle E, <, \sigma\rangle$, wobei $<$ die eine totale Ordnung über $E$ ist.

Damit ist aber die Operation $\cdot$ nicht mehr definiert, die ja auf einer „Indexberechnung“ beruht. Wir geben später eine allgemeinere Definition dieser Operation für beschriftete Halbordnungen an. □

Aus der Theorie der Halbordnungen können wir die Begriffe des Homomorphismus und der Einbettung übernehmen. Eine beschriftete Halbordnung $x$ ist homomorph zu einer beschrifteten Halbordnung $y$, wenn es eine ordnungs- und beschriftungserhaltende Abbildung (d. h. einen *Homomorphismus*) der Ereignismenge von $x$ auf die Ereignismenge von $y$ gibt. Insbesondere sind wir an bijektiven Homomorphismen interessiert, die eine Verdichtung der Kausalordnung beschreiben: Existiert ein solcher bijektiver Homomorphismus von $x$ nach $y$, so enthält $y$ jede Kausalbeziehung, die auch $x$ enthält, aber u. U. noch weitere.

Eine *Einbettung* von $x$ in $y$ ist ein injektiver Homomorphismus von $x$ nach $y$, der für jedes Ereignis $e$ von $x$ die Vorgeschichte $\leqslant_x^{-1}(e)$ von $e$ inklusive aller in ihr auftretenden Kausalbeziehungen auf einen Teil von $y$ abbildet, ohne weitere Kausalbeziehungen in diesem Teil zu erlauben. Offenbar ist eine bijektive Einbettung von $x$ in $y$ ein Isomorphismus auf beschrifteten Halbordnungen, d. h. eine ordnungs- und beschriftungserhaltende Umbenennung der Ereignisse von $x$.

I.2.4. Definition (Homomorphismus und Einbettung). Seien $x$ und $y$ beschriftete Halbordnungen über einem Alphabet $A$. Eine Abbildung $h : E_x \to E_y$ wird als *Homomorphismus* von $x$ nach $y$ bezeichnet, wenn

$$e_1 <_x e_2 \Rightarrow h(e_1) <_y h(e_2)$$

für alle $e_1, e_2 \in E_x$ und weiterhin

$$\lambda_x = \lambda_y \circ h$$

gilt. Ein injektiver Homomorphismus $h$ von $x$ nach $y$ wird *Einbettung* von $x$ in $y$ genannt, wenn

$$h\left(\leqslant_x^{-1}(e)\right) = \leqslant_y^{-1}(h(e))$$

für alle $e \in E_x$ gilt. Eine bijektive Einbettung heißt *Isomorphismus auf beschrifteten Halbordnungen.* □

Da wir beschriftete Halbordnungen als Beschreibung von Verhalten nebenläufiger Systeme verwenden wollen, ist es nützlich, von *Teilverhalten* in der Weise sprechen zu können, daß eine beschriftete Halbordnung $x$ ein Anfangsstück (Präfix) einer beschrifteten Halbordnung $y$ ist. Da wir nicht voraussetzen wollen, daß die Ereignismenge von $x$ eine Teilmenge der Ereignismenge von $y$ ist, verwenden wir hier den Begriff der Einbettung.

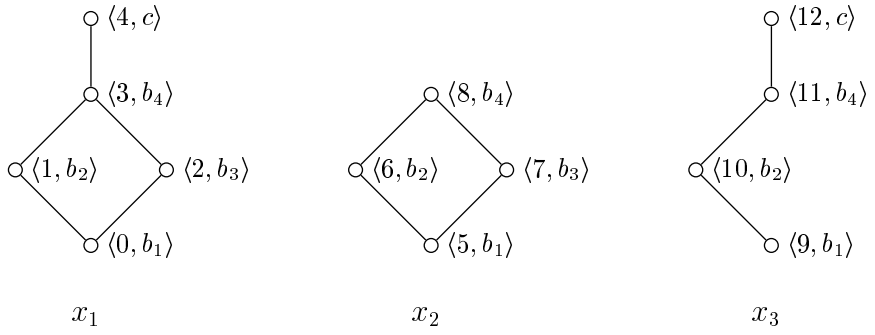



ABBILDUNG I.2. Beispiele für die Präfixrelation

Unterscheiden sich zwei beschriftete Halbordnungen lediglich durch die Dichte ihrer Kausalordnung, können wir diese Halbordnungen bzgl. des Grades an Nebenläufigkeit anordnen; hierfür verwenden wir bijektive Homomorphismen.

I.2.5. DEFINITION (Präfix, Nebenläufigkeit, Isomorphie). Seien $x$ und $y$ beschriftete Halbordnungen über demselben Alphabet.

(a) $x$ wird als *Präfix* von $y$ bezeichnet, wenn es eine Einbettung von $x$ in $y$ gibt. Wir schreiben dann $x \leqslant y$.
(b) $x$ heißt *nebenläufiger als* $y$, wenn es einen bijektiven Homomorphismus von $x$ nach $y$ gibt. In diesem Fall schreiben wir $x \preccurlyeq y$ und nennen $y$ *sequentieller als* $x$
(c) $x$ ist *isomorph zu* $y$, wenn es einen Isomorphismus von $x$ nach $y$ gibt. Wir schreiben $x \equiv y$. Offenbar gilt $x \equiv y$ gdw. $x \preccurlyeq y$ und $y \preccurlyeq x$ gdw. $x \leqslant y$ und $y \leqslant x$ vorliegt.

□

I.2.6. BEISPIEL. Abbildung I.2 zeigt beschriftete Halbordnungen $x_1$, $x_2$ und $x_3$ über dem Alphabet $\{b_1, b_2, b_2, b_4, c\}$. Es gilt $x_2 \leqslant x_1$, die benötigte Einbettung ist

$$h : 5 \mapsto 0, 6 \mapsto 1, 7 \mapsto 2, 8 \mapsto 3$$

Es gibt keine Einbettung von $x_3$ nach $x_1$: Gäbe es eine derartige Abbildung $h$, so müßte wegen $\lambda_{x_1} = \lambda_{x_3} \circ h$

$$h(9) = 0, h(10) = 1, h(11) = 3 \text{ und } h(12) = 4$$

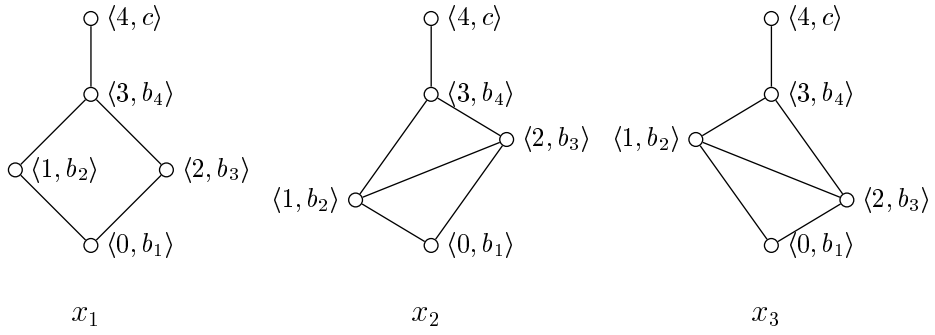



ABBILDUNG I.3. Beispiele für Sequentialisierungen.

gelten. Nun folgt aber

$$\begin{aligned} h(\leqslant_{x_3}^{-1}(11)) &= h(\{9,10,11\}) \\ &= \{0,1,3\} \\ &\neq \{0,1,2,3\} \\ &= \leqslant_{x_1}^{-1}(3) \\ &= \leqslant_{x_1}^{-1}(h(11)). \end{aligned}$$

Abbildung I.3 stellt zwei Möglichkeiten dar, die beschriftete Halbordnung $x_1$ zu sequentialisieren. Es gilt $x_1 \preccurlyeq x_2$, da ein bijektiver Homomorphismus $g : i \mapsto i$ für $0 \leqslant i \leqslant 4$ existiert, ebenso gilt $x_1 \preccurlyeq x_3$ aufgrund der Existenz von $g$.

$x_2 \preccurlyeq x_3$ gilt jedoch nicht: Da etwa die Beziehung $2 \lessdot_{x_2} 1$ vorliegt, müßte ebenso $2 \lessdot_{x_3} 1$ gelten. Das ist jedoch wegen $1 \lessdot_{x_3} 2$ ausgeschlossen. □

I.2.7. DEFINITION. Sei $x$ eine beschriftete Halbordnung über dem Alphabet $A$. Mit

$$\mathbf{C}(x) =_{\text{Def}} \left\{ D \subseteq E_x : D = \leqslant_x^{-1}(D) \right\}$$

bezeichnen wir die Menge all derjenigen Mengen $D \subseteq E_x$, die abgeschlossen bzgl. $\leqslant_x^{-1}$ sind.

Ist $D \subseteq E_x$, so bezeichnet $x[D]$ die beschriftete Halbordnung

$$x[D] =_{\text{Def}} \langle D, <_x \cap (D \times D), \lambda_x \restriction D \rangle.$$

Offenbar gilt $x[D] \leqslant x$ für alle $D \in \mathbf{C}(x)$, die notwendige Einbettung ist $\mathrm{id}_D$. □

Die Menge der Linearisierungen $\mathrm{lin}(x)$ einer beschrifteten Halbordnung $x \in \mathbf{LPO}(A)$ besteht aus beschrifteten Halbordnungen $\sigma$, deren Kausalordnung $<_\sigma$ eine lineare Ordnung ist und die gleichzeitig sequentieller als $x$ sind. Mit Bemerkung I.2.3 ist $\mathrm{lin}(x) \subseteq A^*$.

I.2.8. DEFINITION (Linearisierung). Ist $x \in \mathbf{LPO}(A)$, so bezeichnen wir mit

$$\mathrm{lin}(x) =_{\mathrm{Def}} \{\sigma \in \mathbf{LPO}(A) : x \preccurlyeq \sigma \;\&\; <_\sigma \text{ ist eine lineare Ordnung}\}$$

die Menge der *Linearisierungen* von $x$. Weiterhin setzen wir

$$\mathrm{lin}_x(C) =_{\mathrm{Def}} \mathrm{lin}(x[C])$$

für alle $C \subseteq E_x$. □

I.2.9. LEMMA. *Für alle $x, y \in \mathbf{LPO}(A)$ gilt:*

(a) $x \preccurlyeq y \Rightarrow \mathrm{lin}(x) \supseteq \mathrm{lin}(y)$.
(b) $x \equiv y \Rightarrow \mathrm{lin}(x) = \mathrm{lin}(y)$.

I.2.10. BEMERKUNG. Die Umkehrungen von Lemma I.2.9,(a) und (b) gelten nicht: Ist $x = \langle\{0,1\}, \varnothing, \{\langle 0,a\rangle, \langle 1,a\rangle\}\rangle$ und $y = \langle\{0,1\}, \{\langle 0,1\rangle\}, \{\langle 0,a\rangle, \langle 1,a\rangle\}\rangle$, so gilt $\mathrm{lin}(x) = \mathrm{lin}(y) = \{aa\}$, jedoch weder $x \equiv y$ noch $y \preccurlyeq x$. □

Das folgende Lemma ist einfach zu beweisen:

I.2.11. LEMMA. *$\leqslant$ und $\preccurlyeq$ sind Quasiordnungen über $\mathbf{LPO}(A)$. $\equiv$ ist eine Äquivalenzrelation.*

Der Schlüssel zu dem Beweis von Lemma I.2.11 ist das folgende Lemma:

I.2.12. LEMMA. *Seien $x$, $y$ und $z$ beschriftete Halbordnungen über dem Alphabet $A$.*

(a) *Sind $f : E_x \to E_y$ und $g : E_y \to E_z$ Homomorphismen von $x$ nach $y$ und von $y$ nach $z$, so ist $g \circ f : E_x \to E_z$ ein Homomorphismus von $x$ nach $z$.*
(b) *Insbesondere ist die Komposition $g \circ f : E_x \to E_z$ zweier Einbettungen $f : E_x \to E_y$ und $g : E_y \to E_z$ wiederum eine Einbettung.*
(c) *Ebenso ist die Komposition $g \circ f : E_x \to E_z$ bijektiver Homomorphismen $f : E_x \to E_y$ und $g : E_y \to E_z$ ein bijektiver Homomorphismus.*
(d) $\mathrm{id}_{E_x}$ *ist ein bijektiver Homomorphismen und eine Einbettung von $x$ nach $x$.*

Wir kommen nun zu einer wichtigen Einschränkung der in dieser Arbeit betrachteten beschrifteten Halbordnungen: Wir schließen solche Halbordnungen $x$ aus, die Selbstnebenläufigkeiten enthalten, d. h. Ereignisse, die einerseits gleich beschriftet, andererseits jedoch unabhängig sind. Beschriftete Halbordnungen, die keine derartigen Selbstnebenläufigkeiten enthalten, bezeichnen wir als *Semiordnungen*.

Die Frage, ob eine Semantik für nebenläufige Systeme Selbstnebenläufigkeiten erlauben oder verbieten sollte, behandeln wir in dieser Arbeit nicht weiter. Der in Abschnitt I.3 beschriebene Systembegriff schließt Selbstnebenläufigkeiten von vornherein aus, so daß diese Frage irrelevant wird.

I.2.13. Definition (Semiordnung). Eine *Semiordnung* ist eine beschriftete Halbordnung $x$ mit der folgenden Eigenschaft: Für alle $e_1, e_2 \in E_x$ folgt aus $e_1 \operatorname{co}_x e_2$ bereits $\lambda_x(e_1) \neq \lambda_x(e_2)$. $\mathbf{SO}(A)$ bezeichnet die Klasse aller Semiordnungen über $A$. □

I.2.14. Definition (Halbwörter und Semiwörter). Ein *Halbwort* über einem Alphabet $A$ ist eine Äquivalenzklasse von beschrifteten Halbordnungen über $A$ bzgl. $\equiv$. Ein Semiwort über $A$ ist eine Äquivalenzklasse von Semiordnungen. Wir notieren die Äquivalenzklasse einer beschrifteten Halbordnung durch

$$[x] = [E_x, <_x, \lambda_x] =_{\text{Def}} \{y \in \mathbf{LPO}(A) : x \equiv y\}.$$

Mit $\mathbf{PW}(A)$ bzw. $\mathbf{SW}(A)$ bezeichnen wir die Klassen der Halbwörter bzw. Semiwörter über $A$. Jede Menge $X \subseteq \mathbf{SW}(A)$ wird als *Semisprache* bezeichnet. □

Wir verwenden die folgende Konvention: Sind $x, y, z, \ldots \in \mathbf{LPO}(A)$ beschriftete Halbordnungen, so bezeichnen wir ihre Äquivalenzklassen $[x]$, $[y]$, $[z]$, ... durch kleine Buchstaben in Fettdruck $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$, ... Damit bezeichnet etwa $E_x$ die Menge der Ereignisse eines Repräsentanten eines Halbworts $\boldsymbol{x} = [x]$. Ist $a$ ein Buchstabe, so bezeichnet $\boldsymbol{a}$ das Semiwort

$$[a] =_{\text{Def}} [\{a\}, \varnothing, a \mapsto a].$$

Die Äquivalenzklasse von $\epsilon$ wird mit $\boldsymbol{\epsilon}$ bezeichnet.

I.2.15. Bemerkung (Wörter). In Bemerkung I.2.3 haben wir Sequenzen als totale Semiordnungen aufgefaßt. Die Begriffe „Halbwort“ und „Semiwort“ legen es jedoch nahe, Wörter (nämlich Sequenzen) im Interesse einer konsistenten Begriffsbildung ebenfalls als Äquivalenzklassen zu definieren. Die oben aufgeführten Schreibweisen würden dann jedoch erfordern, Elemente $\sigma \in A^*$ ganz und gar abseits jeder mathematischen Gepflogenheit jeweils in der Form $\boldsymbol{\sigma}$ bzw. $[\sigma]$ zu schreiben. □

I.2.16. Lemma. *Wenn wir*

$$\boldsymbol{x} < \boldsymbol{y} \Leftrightarrow_{Def} x < y \text{ und } \boldsymbol{x} \prec \boldsymbol{y} \Leftrightarrow_{Def} x \prec y$$

*für alle $x \in \boldsymbol{x}$, $y \in \boldsymbol{y}$ setzen, so sind $<$ und $\prec$ Halbordnungen über $\mathbf{PW}(A)$ und $\mathbf{SW}(A)$.*

I.2.17. Lemma. *Sind $x$ und $y$ Semiordnungen über $A$, so gibt es höchstens eine Einbettung $h : E_x \to E_y$.*

Beweis. Seien $h, g : E_x \to E_y$ Einbettungen. Nehmen wir ein $e \in E_x$ mit $h(e) \neq g(e)$ an. Die Annahme $h(e) \operatorname{co}_y g(e)$ führt zu der Berechnung

$$\begin{aligned} h(e) \operatorname{co}_y g(e) &\Rightarrow \lambda_y(h(e)) \neq \lambda_y(g(e)) && (y \text{ Semiordnung}) \\ &\Rightarrow \lambda_x(e) \neq \lambda_x(e), && (\text{Def. Einbettung}) \end{aligned}$$

was unmöglich ist.

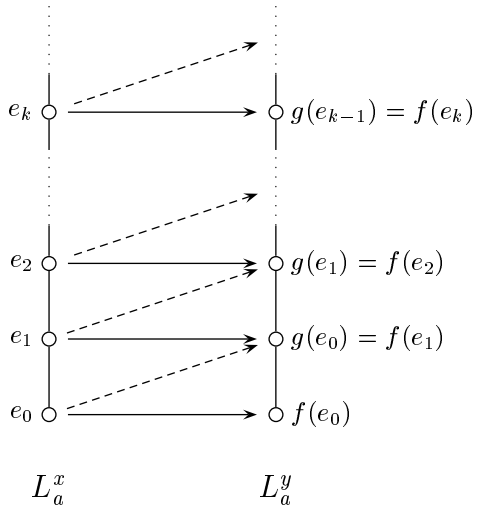



ABBILDUNG I.4. Illustration zum Beweis von Lemma I.2.19.

Nehmen wir deshalb $h(e) <_y g(e)$ an. Daraus folgt $\leqslant_y^{-1}(h(e)) \subset \leqslant_y^{-1}(g(e))$. Da $h$ und $g$ Einbettungen sind, schließen wir $h\,(\leqslant_x^{-1}(e)) \subset g\,(\leqslant_x^{-1}(e))$. Mit der Injektivität von $h$ und $g$ erhalten wir jedoch $|\leqslant_x^{-1}(e)| < |\leqslant_x^{-1}(e)|$. Ebenso führt die Annahme $g(e) <_y h(e)$ zu einem Widerspruch.

Wir schließen $h(e) = g(e)$ für alle $e \in E_x$, d. h. $h = g$. □

Mit Lemma I.2.17 können wir definieren:

I.2.18. DEFINITION (Präfixeinbettung). Seien $x$ und $y$ Semiordnungen über dem Alphabet $A$, so daß $x \leqslant y$ gilt. Dann bezeichnet $H_x^y : E_x \to E_y$ die *Präfixeinbettung* genannte eindeutige Einbettung von $x$ in $y$, □

I.2.19. LEMMA. *Sind $x$ und $y$ Semiordnungen über $A$, so gibt es höchstens einen bijektiven Homomorphismus $g : E_x \to E_y$.*

BEWEIS. Abb. I.4 zeigt eine Illustration der folgenden Konstruktion (durchgezogene Pfeile repräsentieren die Funktion $f$, während $g$ durch gestrichelte Pfeile dargestellt wird).

Nehmen wir zwei verschiedenene bijektive Homomorphismen $f, g : E_x \to E_y$ an, d. h. es gibt ein $e_0 \in E_x$ mit $f(e_0) \neq g(e_0)$. Wegen $\lambda_x = \lambda_y \circ f = \lambda_y \circ g$ gilt $\lambda_x(e_0) = \lambda_y(f(e_0)) = \lambda_y(g(e_0))$. Nehmen wir $\lambda_x(e_0) = a$ an.

Sei $L_a^x = \lambda_x^{-1}(a)$ und $L_a^y = \lambda_y^{-1}(a)$. Aus der Existenz bijektiver Homomorphismen folgt $|L_a^x| = |L_a^y|$, und da sowohl $x$ als auch $y$ Semiordnungen sind, sind beide Mengen Ketten in $x$ bzw. $y$. Insbesondere gilt $e_0 \in L_a^x$ und $f(e_0), g(e_0) \in L_a^y$. O. B. d. A. nehmen wir $f(e_0) <_y g(e_0)$ an.

Sei nun $e_1 \in E_x$ das Urbild von $g(e_0)$ unter $f$, d. h. $g(e_0) = f(e_1)$. Natürlich ist $e_1 \in L_a^x$. Mit $f(e_0) <_y f(e_1)$ und $e_0 \ \mathrm{li}_x \ e_1$ erhalten wir $e_0 <_x e_1$, da $e_1 <_x e_0$ $f(e_1) <_y f(e_0)$ implizieren würde.

Bilden wir nun $e_1$ wiederum mit Hilfe von $g$ auf $g(e_1)$ ab, so erhalten wir mit einer analogen Argumentation $g(e_0) <_y g(e_1)$. Wiederholen wir diese Konstruktion, finden wir ein $e_2$ mit $g(e_1) = f(e_2)$ und $e_1 <_x e_2$, und $g(e_1) <_y g(e_2)$; i. Allg. ist es also möglich, unendliche Ketten $e_0 <_x e_1 <_x \cdots$ in $L_a^x$ und $g(e_0) <_y g(e_1) <_y \cdots$ in $L_a^y$ zu finden. Dies widerspricht aber der Endlichkeit von $E_x$ bzw. $E_y$. □

I.2.20. Bemerkung. Betrachtet man unendliche Semiordnungen, kann die Konstruktion von Ketten in $L_a^x$ bzw. $L_a^y$ im Beweis von Lemma I.2.19 auch „nach unten“ vorgenommen werden, indem man im ersten Schritt $g(e_0) <_y f(e_0)$ annimmt und dann die Existenz unendlicher Ketten $e_0 >_x e_1 >_x \cdots$ und $g(e_0) >_y g(e_1) >_y \cdots$ herleitet. □

I.2.21. Definition (Sequentialisierung). Gilt $x \preccurlyeq y$ für Semiordnungen $x$ und $y$ über dem Alphabet $A$, so bezeichnen wir den nach Lemma I.2.19 eindeutig bestimmten bijektiven Homomorphismus von $x$ nach $y$ mit $G_x^y$; diesen Homomorphismus bezeichnen wir auch als *Sequentialisierung* von $x$ nach $y$. □

I.2.22. Bemerkung. Weder Lemma I.2.17 noch Lemma I.2.19 gelten für beschriftete Halbordnungen, wie das folgende Beispiel zeigt: Ist $x = \langle \{0,1\}, \varnothing, \{\langle 0, a\rangle, \langle 1, a\rangle\}\rangle$, so sind die beiden Abbildungen $f, g : E_x \to E_x$ mit $f : 0 \mapsto 0, 1 \mapsto 1$ und $g : 0 \mapsto 1, 1 \mapsto 0$ Einbettungen und bijektiven Homomorphismen. □

**Operationen auf Semiwörtern.** Starke [**78**] definiert die folgenden Operationen auf Semiwörtern:

(a) *Konkatenation.* Seien $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{SW}(A)$ Semiwörter, so daß $E_x \cap E_y = \varnothing$ für ausgewählte Repräsentanten $x$ und $y$ gilt. Die Konkatenation von $\boldsymbol{x}$ und $\boldsymbol{y}$ ist als

$$\boldsymbol{x} \cdot \boldsymbol{y} \quad =_{\mathrm{Def}} \quad [E_x \cup E_y, <_x \cup <_y \cup (E_x \times E_y), \lambda_x \cup \lambda_y]$$

definiert. Wir schreiben $\boldsymbol{xy}$ anstelle von $\boldsymbol{x} \cdot \boldsymbol{y}$.

(b) *Parallelprodukt.* Seien $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{SW}(A)$ Semiwörter, so daß $E_x \cap E_y = \varnothing$ für ausgewählte Repräsentanten $x$ und $y$ gilt. Ihr Parallelprodukt ist

$$\boldsymbol{x} \times \boldsymbol{y} \quad =_{\mathrm{Def}} \quad \begin{cases} [E_x \cup E_y, <_x \cup <_y, \lambda_x \cup \lambda_y], \\ \qquad \text{falls } \lambda(x) \cap \lambda(y) = \varnothing; \\ \text{undefiniert sonst.} \end{cases}$$

(c) Die Definition einer dritten Operation, $\cup$ (*Komposition*) erfordert den Begriff der *kanonischen Semiordnung*, der ebenfalls in [**78**] eingeführt wird. Für unsere Arbeit ist jedoch eine alternative Definition dieses Begriffs

geeigneter. Da die $\cup$-Operation im folgenden nicht weiter benötigt wird, führen wir ihre Definition nicht auf.[2]

Wir schreiben $x \cdot y$ bzw. $xy$, um einen Repräsentanten von $\boldsymbol{xy}$ zu bezeichnen. Ebenso steht $x \times y$ für einen Repräsentanten von $\boldsymbol{x} \times \boldsymbol{y}$.

I.2.23. Lemma (Starke [**78**]). *Für $\cdot$ gilt:*

(a) $\boldsymbol{xy} \in \mathbf{SW}(T)$,
(b) $\boldsymbol{x}(\boldsymbol{yz}) = (\boldsymbol{xy})\boldsymbol{z}$,
(c) $\boldsymbol{xy} = \boldsymbol{x} \Rightarrow \boldsymbol{y} = \boldsymbol{\epsilon}$,
(d) $\boldsymbol{xy} = \boldsymbol{y} \Rightarrow \boldsymbol{x} = \boldsymbol{\epsilon}$.

*Wenn die genannten Teilausdrücke definiert sind, gilt für $\times$:*

(e) $\boldsymbol{x} \times \boldsymbol{y} \in \mathbf{SW}(T)$,
(f) $\boldsymbol{x} \times (\boldsymbol{y} \times \boldsymbol{z}) = (\boldsymbol{x} \times \boldsymbol{y}) \times \boldsymbol{z}$,
(g) $\boldsymbol{x} \times \boldsymbol{y} = \boldsymbol{y} \times \boldsymbol{x}$,
(h) $\boldsymbol{x} \times \boldsymbol{y} = \boldsymbol{y} \Rightarrow \boldsymbol{x} = \boldsymbol{\epsilon}$.

I.2.24. Definition (Schrittrelation). Für alle $z \in \mathbf{SO}(A)$ ist die *Schrittrelation* $\overset{z}{\Longrightarrow} \subseteq \mathbf{SO}(A) \times \mathbf{SO}(A)$ wie folgt definiert:

$$x \overset{z}{\Longrightarrow} y \Leftrightarrow_{\mathrm{Def}} x \leqslant y \;\&\; z \equiv y\,[E_y - H_x^y(E_x)]\,.$$

Da diese Operation offenbar invariant bzgl. $\equiv$ ist, setzen wir

$$\boldsymbol{x} \overset{\boldsymbol{z}}{\Longrightarrow} \boldsymbol{y} \Leftrightarrow_{\mathrm{Def}} x \overset{z}{\Longrightarrow} y$$

für alle $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbf{SW}(A)$. □

**Kanonische Semiordnungen.** Semiwörter wurden als Äquivalenzklassen von Semiordnungen bzgl. Isomorphie definiert. Es stellt sich die Frage, ob eine *kanonische Darstellung* von Semiordnungen existiert. Eine solche Darstellung erfolgt für zwei Semiordnungen genau dann gleich, wenn diese isomorph sind. Offenbar muß die kanonische Darstellung einer Semiordnung $x$ unabhängig von der Benennung (nicht der Beschriftung!) der Ereignisse von $x$ sein.

Eine solche Darstellung kann auf verschiedene Weise gewonnen werden. Starke [**78**] verwendet zur Definition einer kanonischen Semiordnung $x$ über dem Alphabet $A$ die Abbildung $\eta_x : e \mapsto \langle \lambda_x(e), n\rangle \in A \times \mathbb{N}$ für alle $e \in E_x$, wobei

$$n = \max_{\leqslant} \left\{ \left| L \cap \lambda_x^{-1}(\lambda_x(e)) \right| : L \text{ Kette in } \leqslant_x^{-1}(e) \right\}$$

ist; die Semiordnung $\langle \eta_x(E_x), <, \langle a, n\rangle \mapsto a\rangle$ mit $\eta_x(e) < \eta_x(e') \Leftrightarrow_{\mathrm{Def}} e <_x e'$ ist dann kanonisch in $\boldsymbol{x}$. Der Nachteil dieser Vorgehensweise ist, daß die Kausalordnung einer kanonischen Semiordnung nicht direkt aus der Kodierung der Ereignisse

[2] Allerdings kann jedes Element der Klasse $\mathbf{SW}(A)$ durch die iterierte Anwendung der Operationen $\cdot$ und $\cup$ auf Buchstaben beschrieben werden; $\mathbf{SW}(A)$ ist jedoch nicht abgeschlossen bzgl. der Anwendung von $\cdot$ und $\times$.

dieser Semiordnung hergeleitet werden kann (vgl. Äquivalenz (I.2.$\beta$) in Definition I.2.25).

Wir benutzen die Abbildung $\gamma_x : e \mapsto \langle \lambda_x(e), \gamma_x\,(<_x^{-1}(e))\rangle$, damit ähnelt unser Ansatz dem von Engelfriet [**29**] zur kanonischen Darstellung verzweigter Abläufe von Petri-Netzen, allerdings verwendet Engelfriet Funktionen der Form $\gamma'_x : e \mapsto \langle \lambda_x(e), \gamma'_x\,(\lessdot_x^{-1}(e))\rangle$. Engelfriets Definition führt zu einer kompakteren Kodierung der Ereignisse einer Semiordnung, der Umgang mit ihr ist jedoch teilweise komplizierter.

I.2.25. Definition (Kanonische Semiordnung). Für ein Alphabet $A$ bezeichnen wir mit $\Gamma(A)$ die kleinste Menge, die die Gleichung $X = A \times \mathscr{P}_{\mathrm{f}}(X) \cup \{\varnothing\}$ in $X$ erfüllt. Ist $x \in \mathbf{SO}(A)$, so definieren wir die Abbildung $\gamma_x : E_x \to \Gamma(A)$ rekursiv als

$$\gamma_x(e) \quad =_{\mathrm{Def}} \quad \big\langle \lambda_x(e), \gamma_x\left(<_x^{-1}(e)\right)\big\rangle.$$

Nun sei $\gamma(x) = \big\langle E_{\gamma(x)}, <_{\gamma(x)}, \lambda_{\gamma(x)}\big\rangle$ eine Struktur mit den Komponenten

$$E_{\gamma(x)} \quad =_{\mathrm{Def}} \quad \gamma_x(E_x), \tag{I.2.$\alpha$}$$

$$\gamma_x(e_1) <_{\gamma(x)} \gamma_x(e_2) \quad \Leftrightarrow_{\mathrm{Def}} \quad \gamma_x(e_1) \in \gamma_x\left(<_x^{-1}(e_2)\right), \text{ und} \tag{I.2.$\beta$}$$

$$\lambda_{\gamma(x)}(\gamma_x(e)) \quad =_{\mathrm{Def}} \quad \lambda_x(e). \tag{I.2.$\gamma$}$$

$\gamma(x)$ wird die *kanonische Semiordnung* zu $x$ genannt. □

I.2.26. Bemerkung. $\gamma(x)$ ist tatsächlich eine Semiordnung. Dies folgt aus Theorem I.2.29, (a) und (b) sowie Gleichung (I.2.$\gamma$). □

I.2.27. Bemerkung. $\Gamma(A)$ kann als die Menge der endlichen Bäume aufgefaßt werden, deren Knoten mit Elementen aus $A$ beschriftet sind. Um den Aufbau von $\Gamma(A)$ zu verdeutlichen, geben wir eine alternative Konstruktion an.

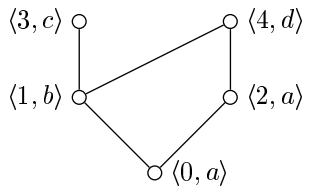
Wir definieren

$$\begin{aligned}
\Gamma^0(A) &=_{\mathrm{Def}} \{\varnothing\},\\
\Gamma^n(A) &=_{\mathrm{Def}} A \times \mathscr{P}_{\mathrm{f}}\left(\bigcup_{m<n} \Gamma^m(A)\right) \text{ für } n > 0, \text{ sowie}\\
\Gamma(A) &=_{\mathrm{Def}} \bigcup_{n\geqslant 0} \Gamma^n(A).
\end{aligned}$$

Eine Standardargumentation erlaubt leicht den Nachweis, daß $\Gamma(A)$ wirklich die kleinste Menge mit $\Gamma(A) = A \times \mathscr{P}_{\mathrm{f}}(\Gamma(A)) \cup \{\varnothing\}$ ist. Zur Illustration berechnen wir die ersten drei Glieder der Iteration für $A = \{a, b\}$:

$$\begin{aligned}
&\Gamma^0(A) = \{\varnothing\}, \qquad \Gamma^1(A) = \{\langle a, \varnothing\rangle, \langle b, \varnothing\rangle\},\\
&\Gamma^2(A) = \{\langle a, \varnothing\rangle, \langle b, \varnothing\rangle, \langle a, \{\langle a, \varnothing\rangle\}\rangle, \langle a, \{\langle b, \varnothing\rangle\}\rangle, \langle b, \{\langle a, \varnothing\rangle\}\rangle, \langle b, \{\langle b, \varnothing\rangle\}\rangle,\\
&\qquad \langle a, \{\langle a, \varnothing\rangle, \langle b, \varnothing\rangle\}\rangle, \langle b, \{\langle a, \varnothing\rangle, \langle b, \varnothing\rangle\}\rangle\}.
\end{aligned}$$

□




$x$

ABBILDUNG I.5. Semiordnung zur Illustration des Begriffs der kanonischen Semiordnung.

---

I.2.28. BEISPIEL (Kanonische Semiordnung). Betrachten wir die Semiordnung in Abbildung I.5 über $A = \{a, b, c, d\}$. Die Funktion $\gamma_x$ bildet die Ereignisse von $x$ wie folgt auf Elemente aus $\Gamma(A)$ ab:

$$\begin{array}{lcl}
\gamma_x(0) & = & \langle a, \varnothing\rangle, \\
\gamma_x(1) & = & \langle b, \{\langle a, \varnothing\rangle\}\rangle, \\
\gamma_x(2) & = & \langle a, \{\langle a, \varnothing\rangle\}\rangle, \\
\gamma_x(3) & = & \langle c, \{\langle b, \{\langle a, \varnothing\rangle\}\rangle, \langle a, \varnothing\rangle\}\rangle, \\
\gamma_x(4) & = & \langle d, \{\langle b, \{\langle a, \varnothing\rangle\}\rangle, \langle a, \{\langle a, \varnothing\rangle\}\rangle, \langle a, \varnothing\rangle\}\rangle.
\end{array}$$

□

I.2.29. THEOREM. *Für alle Semiordnungen $x, y$ über $A$ gilt:*

(a) *$\gamma_x$ ist injektiv;*
(b) *$e_1 <_x e_2$ gdw. $\gamma_x(e_1) <_{\gamma(x)} \gamma_x(e_2)$;*
(c) *$\gamma_x$ ist eine Einbettung;*
(d) *$x \leqslant y$ gdw. $E_{\gamma(x)} \subseteq E_{\gamma(y)}$;*
(e) *$x \equiv y$ gdw. $\gamma(x) = \gamma(y)$.*

BEWEIS. (a) Induktion über $<_x$. Seien $e_1, e_2 \in \min(x)$. Dann berechnen wir

$$\begin{array}{rcl}
\gamma_x(e_1) = \gamma_x(e_2) & \Rightarrow & \langle \lambda_x(e_1), \varnothing\rangle = \langle \lambda_x(e_2), \varnothing\rangle \\
& \Rightarrow & \lambda_x(e_1) = \lambda_x(e_2) \\
& \Rightarrow & e_1 = e_2,
\end{array}$$

da $e_1$ $\mathrm{co}_x$ $e_2$ schon $\lambda_x(e_1) \neq \lambda_x(e_2)$ implizieren würde. Als Induktionshypothese nehmen wir Ereignisse $e_1, e_2 \in E_x - \min(x)$ an, so daß die Implikation

$$\gamma_x\left(<_x^{-1}(e_1)\right) = \gamma_x\left(<_x^{-1}(e_2)\right) \Rightarrow <_x^{-1}(e_1) = <_x^{-1}(e_2)$$

---

gilt. Wir berechnen

$$\begin{aligned}\gamma_x(e_1) = \gamma_x(e_2) &\Rightarrow \left\langle \lambda_x(e_1), \gamma_x\left(<_x^{-1}(e_1)\right)\right\rangle = \left\langle \lambda_x(e_2), \gamma_x\left(<_x^{-1}(e_2)\right)\right\rangle \\ &\Rightarrow \lambda_x(e_1) = \lambda_x(e_2) \;\&\; <_x^{-1}(e_1) = <_x^{-1}(e_2) \quad \text{(Induktionshypothese)} \\ &\Rightarrow e_1 = e_2,\end{aligned}$$

da $<_x^{-1}(e_1) \subset <_x^{-1}(e_2)$ aus $e_1 <_x e_2$ folgt und $e_1 \operatorname{co}_x e_2$ einen Widerspruch zu $\lambda_x(e_1) = \lambda_x(e_2)$ darstellen würde.

(b)

$$\begin{aligned}e_1 <_x e_2 &\Leftrightarrow e_1 \in <_x^{-1}(e_2) \\ &\Leftrightarrow \gamma_x(e_1) \in \gamma_x\left(<_x^{-1}(e_2)\right) && \text{(nach (a))} \\ &\Leftrightarrow \gamma_x(e_1) <_{\gamma(x)} \gamma_x(e_2). && \text{(nach (I.2.}\beta\text{))}\end{aligned}$$

(c) Die Injektivität von $\gamma_x$ ist mit (a) bereits gezeigt. Mit der Beobachtung $\lambda_x = \lambda_{\gamma(x)} \circ \gamma_x$ folgt, daß $\gamma_x$ ein Homomorphismus ist. Somit ist nur noch

$$\gamma_x\left(\leqslant_x^{-1}(e)\right) \;=\; \leqslant_{\gamma(x)}^{-1}(\gamma_x(e))$$

nachzuweisen. Diese Äquivalenz folgt jedoch mit Hilfe einer einfachen Induktion über $\leqslant_x$, wie wir sie bereits in ähnlicher Form für den Beweis von (a) vorgenommen haben.

(d, $\Rightarrow$) Wenn $x \leqslant y$ gilt, gibt es die injektive Einbettung $H_x^y$. Nach (c) sind $\gamma_x$ und $\gamma_y$ injektive Einbettungen, also ist $\gamma_y \circ H_x^y : E_x \to E_{\gamma(y)}$ eine injektive Einbettung nach Lemma I.2.12. Weiterhin gilt nach Lemma I.2.17 $\gamma_x = \gamma_y \circ H_x^y$. Es folgt

$$\begin{aligned}\gamma_x(e) \in E_{\gamma(x)} &\Rightarrow \gamma_y(H_x^y(e)) \in E_{\gamma(x)} \\ &\Rightarrow \gamma_y(H_x^y(e)) \in E_{\gamma(y)}.\end{aligned}$$

(d, $\Leftarrow$) Nehmen wir nun $E_{\gamma(x)} \subseteq E_{\gamma(y)}$ an. Dann existiert offenbar eine injektive Einbettung $i : E_{\gamma(x)} \to E_{\gamma(y)}$, die durch $i(e) =_{\text{Def}} e$ für alle $e \in E_{\gamma(x)}$ definiert ist. Nach (c) ist $\gamma_y^{-1} \circ i \circ \gamma_x : E_x \to E_y$ ebenfalls eine injektive Einbettung.

(e) folgt direkt aus (d). □

**Durchschnitt und Vereinigung.** Der Durchschnitt zweier Semiwörter $\boldsymbol{x}$ und $\boldsymbol{y}$ ist als ihr größter gemeinsamer Präfix definiert, d. h. ihre größte untere Schranke bzgl. $\leqslant$. Ihre Vereinigung ist das kleinste $\boldsymbol{z}$, so daß sowohl $\boldsymbol{x}$ als auch $\boldsymbol{y}$ Präfixe von $\boldsymbol{z}$ sind; $\boldsymbol{z}$ ist also die kleinste obere Schranke von $\boldsymbol{x}$ und $\boldsymbol{y}$.

I.2.30. Definition (Durchschnitt). Sei $X \subseteq \mathbf{SW}(A)$ eine nicht leere Menge von Semiwörtern. Ein *gemeinsamer Präfix* von $X$ ist ein Semiwort $\boldsymbol{x}$ über $A$, so daß $\boldsymbol{x} \leqslant \boldsymbol{y}$ für alle $\boldsymbol{y} \in X$ gilt. Wenn $X$ einen bzgl. $\leqslant$ maximalen gemeinsamen Präfix hat, wird dieser *Durchschnitt* von $X$ genannt und mit $\bigwedge X$ bezeichnet. Ist $X = \{\boldsymbol{x}, \boldsymbol{y}\}$, so schreiben wir $\boldsymbol{x} \wedge \boldsymbol{y}$. □

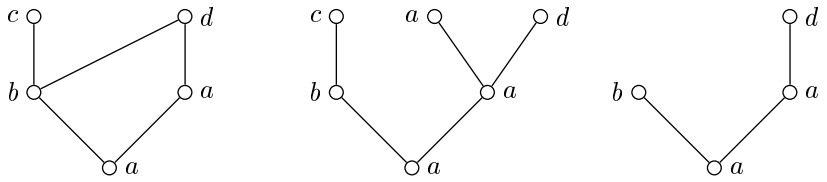



ABBILDUNG I.6. Durchschnitt zweier Semiwörter.

I.2.31. BEISPIEL (Durchschnitt). Abbildung I.6 zeigt den Durchschnitt $\boldsymbol{z}$ der Semiwörter $\boldsymbol{x}$ und $\boldsymbol{y}$. Wir stellen Semiwörter ebenso wie Semiordnungen als Hassediagramme dar, wobei wir aber nur die Ereignisbeschriftungen zeigen. □

Offenbar hat jede nicht leere Menge von Semiwörtern einen gemeinsamen Präfix, nämlich $\boldsymbol{\epsilon}$. Das folgende Lemma zeigt, daß jede nicht leere Menge von Semiwörtern ebenfalls einen eindeutig bestimmten Durchschnitt hat.

I.2.32. LEMMA. *Ist $X \subseteq \mathbf{SW}(A)$, dann existiert ein eindeutiger Durchschnitt $\bigwedge X \in \mathbf{SW}(A)$.*

BEWEIS. Wir definieren

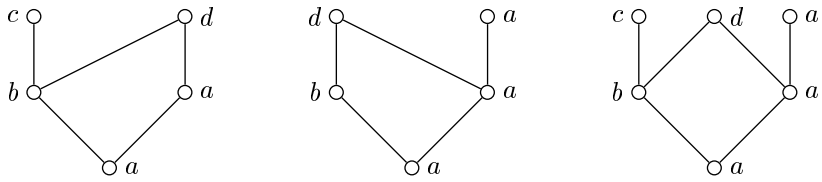
$$z \quad =_{\text{Def}} \quad \left\langle \bigcap_{\boldsymbol{x}\in X} E_{\gamma(x)}, \bigcap_{\boldsymbol{x}\in X} <_{\gamma(x)}, \bigcap_{\boldsymbol{x}\in X} \lambda_{\gamma(x)} \right\rangle.$$

Da der Durchschnitt von Halbordnungen ebenfalls eine Halbordnung ist, ist $z$ eine beschriftete Halbordnung. Offenbar gibt es für jede unabhängige Menge $C \subseteq E_z$ eine unabhängige Menge $C' \in E_{\gamma(x)}$ für ein $\boldsymbol{x} \in X$, so daß $C \subseteq C'$ gilt. Dies impliziert, daß $z$ eine Semiordnung ist. $z = \gamma(z)$ ist leicht zu verifizieren, $z$ ist also eine kanonische Semiordnung. Weiterhin beobachten wir, daß $\bigcap_{\boldsymbol{x}\in X} E_{\gamma(x)} \subseteq E_{\gamma(x)}$ für alle $\boldsymbol{x} \in X$ gilt. Damit ist nach Theorem I.2.29.(d) $\boldsymbol{z}$ ein gemeinsamer Präfix von $X$.

Sei nun $\boldsymbol{y}$ ein weiterer gemeinsamer Präfix von $X$. Durch eine weitere Anwendung von Theorem I.2.29.(d) erhalten wir $E_{\gamma(y)} \subseteq E_{\gamma(x)}$ für alle $\boldsymbol{x} \in X$, da $E_{\gamma(y)} \subseteq \bigcap_{\boldsymbol{x}\in X} E_{\gamma(x)} = E_z$ gilt. Wir folgern $\boldsymbol{y} \leqslant \boldsymbol{z}$, d. h. $\boldsymbol{z} = \bigwedge X$. □

I.2.33. LEMMA. *Seien $\boldsymbol{x}$, $\boldsymbol{y}$, und $\boldsymbol{z}$ Semiordnungen über demselben Alphabet. Die folgenden Gesetze sind erfüllt:*

(a) $\boldsymbol{x} \wedge \boldsymbol{y} = \boldsymbol{y} \wedge \boldsymbol{x}$,
(b) $\boldsymbol{x} \wedge \boldsymbol{\epsilon} = \boldsymbol{\epsilon} \wedge \boldsymbol{x} = \boldsymbol{\epsilon}$,
(c) $\boldsymbol{x} \wedge (\boldsymbol{y} \wedge \boldsymbol{z}) = (\boldsymbol{x} \wedge \boldsymbol{y}) \wedge \boldsymbol{z}$,
(d) $\boldsymbol{x} \leqslant \boldsymbol{y} \Leftrightarrow \boldsymbol{x} = \boldsymbol{x} \wedge \boldsymbol{y}$.




$\boldsymbol{x}$ $\boldsymbol{y}$ $\boldsymbol{z} = \boldsymbol{x} \vee \boldsymbol{y}$

ABBILDUNG I.7. Vereinigung zweier Semiwörter

---

BEWEIS. $\boldsymbol{x} \wedge \boldsymbol{y}$ ist eine größte untere Schranke von $\boldsymbol{x}$ und $\boldsymbol{y}$. Damit folgen (a) bis (d) aus den Theoremen der Verbandstheorie (siehe z. B. [**10**], Lemma 1). □

I.2.34. DEFINITION (Vereinigung). Mit $\bigvee X$ notieren wir die *Vereinigung* einer Menge von Semiwörten $X$, die durch

$$\bigvee X \quad =_{\text{Def}} \quad \left[ \bigcup_{\boldsymbol{x} \in X} E_{\gamma(x)}, \bigcup_{\boldsymbol{x} \in X} <_{\gamma(x)}, \bigcup_{\boldsymbol{x} \in X} \lambda_{\gamma(x)} \right].$$

gegeben ist. Diese Operation ist nur dann definiert, wenn $\bigvee X$ ein Semiwort ist. Wiederum steht $\boldsymbol{x} \vee \boldsymbol{x}$ für $\bigvee\{\boldsymbol{x}, \boldsymbol{y}\}$. □

I.2.35. BEISPIEL. Abbildung I.7 zeigt die Vereinigung $\boldsymbol{z}$ zweier Semiwörter $\boldsymbol{x}$ und $\boldsymbol{y}$.

Die komponentenweise Vereinigung der kanonischer Repräsentanten der Semiwörter $\boldsymbol{ba}$ und $\boldsymbol{ca}$ ist allerdings kein Semiwort; die aus der Anwendung der Berechnungsvorschrift resultierende Struktur (ein Halbwort) ist

$$\big[\{0,1,2,3\}, \{\langle 0,1\rangle, \langle 2,3\rangle\}, \{\langle 0,b\rangle, \langle 1,a\rangle, \langle 2,c\rangle, \langle 3,a\rangle\}\big].$$

Ein anderes Beispiel für eine Menge von Semiwörtern, deren Vereinigung nicht definiert ist, ist die *ω-Kette* $X = \{\boldsymbol{a}^n : n \geqslant 0\}$ mit

$$\boldsymbol{a}^n =_{\text{Def}} \underbrace{\boldsymbol{a}\boldsymbol{a}\ldots\boldsymbol{a}}_{n\text{-mal}}$$

Eine kleinste obere Schranke dieser Menge ist nämlich $[\mathbb{N}, <, n \mapsto a]$ (wobei wir stillschweigend den Isomorphiebegriff auf unendliche Semiordnungen erweitert haben). □

I.2.36. LEMMA. *Wenn $\bigvee X$ ein Semiwort ist, so ist $\bigvee X$ eine kleinste obere Schranke bzgl. $\leqslant$ von $X$.*

BEWEIS. Analog zum Beweis von Lemma I.2.32. □

I.2.37. LEMMA. *Seien $\boldsymbol{x}$, $\boldsymbol{y}$ und $\boldsymbol{z}$ Semiwörter über demselben Alphabet. Die folgenden Gesetze sind erfüllt, wenn die genannten Ausdrücke existieren.*

---

(a) $\boldsymbol{x} \vee \boldsymbol{y} = \boldsymbol{y} \vee \boldsymbol{x}$,
(b) $\boldsymbol{x} \vee \boldsymbol{\epsilon} = \boldsymbol{\epsilon} \vee \boldsymbol{x} = \boldsymbol{x}$ *(immer definiert)*,
(c) $\boldsymbol{x} \vee (\boldsymbol{y} \vee \boldsymbol{z}) = (\boldsymbol{x} \vee \boldsymbol{y}) \vee \boldsymbol{z}$,
(d) $\boldsymbol{x} \wedge \boldsymbol{y} \leqslant \boldsymbol{x} \vee \boldsymbol{y}$,
(e) $\boldsymbol{x} \leqslant \boldsymbol{y} \Leftrightarrow \boldsymbol{y} = \boldsymbol{x} \vee \boldsymbol{y}$,
(f) $\boldsymbol{x} \wedge (\boldsymbol{x} \vee \boldsymbol{y}) = \boldsymbol{x} \vee (\boldsymbol{x} \wedge \boldsymbol{y}) = \boldsymbol{x}$,
(g) $(\boldsymbol{x} \wedge \boldsymbol{y}) \vee \boldsymbol{z} = (\boldsymbol{x} \wedge \boldsymbol{z}) \vee (\boldsymbol{y} \wedge \boldsymbol{z})$,
(h) $(\boldsymbol{x} \wedge \boldsymbol{y}) \vee \boldsymbol{z} = (\boldsymbol{x} \vee \boldsymbol{z}) \wedge (\boldsymbol{y} \vee \boldsymbol{z})$.

Beweis. (a) bis (f): [**10**], Lemma 1. Allerdings ist der Rückgriff auf die Verbandstheorie nicht wirklich notwendig, da die Operationen $\vee$ und $\wedge$ auf die Mengenoperationen $\cup$ und $\cap$ zurückgeführt werden können: Die Gesetze gelten im Verband der Teilmengen einer Menge. Diese Beobachtung beweist auch (g) und (h), diese Gesetze gelten nicht für beliebige Halbordnungen. □

I.2.38. Bemerkung. Die Operationen $\wedge$ und $\vee$ eröffnen die Möglichkeit, Mengen von Semiwörtern als Scottsche Bereiche (d. h. algebraische gerichtete vollständige Halbordnungen) in Sinne der Bereichstheorie aufzufassen [**1, 36**]. Allerdings bildet etwa $\mathbf{SW}(A)$ im Gegensatz zu den in [**45**] diskutierten potentiell unendlichen Abläufen selbst noch keinen Scottschen Bereich (vgl. Beispiel I.2.35); wir können jedoch eine Vervollständigungstechnik (etwa die klassische *Idealvervollständigung*) anwenden. Auch die Verwendung unendlicher Semiwörter würde zu einem Scottschen Bereich führen. □

## I.3. Prozesse nebenläufiger Systeme

In diesem Abschnitt werden wir Transitions- und Spursysteme als allgemeines Modell nebenläufiger Systeme einführen. Wir betrachten diskrete Systeme: Diskretheit bedeutet, daß ein Begriff für Systemzustände sowie ein Begriff für atomare Systemaktionen, die Zustände in andere Zustände überführen, existiert. Darüberhinaus setzen wir die Existenz eines *Anfangszustands* voraus.

Nebenläufigkeit ist in solchen *Transitonssystemen* begrifflich nicht enthalten; wir verwenden in dieser Arbeit einen *a-priori* Begriff für das unabhängige Stattfinden von Systemaktionen in Form einer Unabhängigkeitsrelation, die für Systemaktionen (nicht für ihr Stattfinden) definiert ist; ein Transitionssystem, das mit einer solchen Relation versehen wird, nennen wir ein *Spursystem*. Eine von konzessionierenden Systemzuständen unabhängige Unabhängigkeitsrelation bedeutet eine nicht unwesentliche Einschränkung: In manchen Formalismen wie etwa in allgemeinen Platz-Transitionsnetzen und vielen höheren Petri-Netzklassen ist eine vom jeweiligen Systemzustand abhängige Unabhängigkeitsrelation zur adäquaten Beschreibung von Nebenläufigkeit erforderlich.

Allerdings erlaubt die Inkaufnahme dieser Einschränkung eine elegante und konsistente mathematische Handhabung des Prozeßbegriffs für nebenläufige Systeme; nicht zuletzt können wir auf die reichhaltige Theorie der *Mazurkiewiczspuren* zurückgreifen. Darüberhinaus kann eine große Klasse von Formalismen zur Beschreibung nebenläufiger Systeme auf diese Weise behandelt werden.

**Transitionssysteme.** Wir beginnen mit dem Begriff des Transitionssystems. Wir verwenden Petri-Netze (genauer: eine Variante elementarer Netzsysteme) als „Syntax“ für solche Systeme

I.3.1. Definition (Transitionssystem). Ein *Transitionssystem* über einem Alphabet $A$ ist eine Struktur $\mathcal{T} = \langle S, \delta, s\rangle$ mit den Komponenten

(a) $S$, einer Menge von *Zuständen*,
(b) $\delta$, einer Abbildung $S \times A \to \mathcal{P}_{\mathrm{f}}(S)$, die als *Transitionsfunktion* oder *Schaltfunktion* bezeichnet wird,
(c) $s \in S$, einem *initialem Zustand*.

Wir verwenden auch die relationale Schreibweise

$$s_1 \xrightarrow[\mathcal{T}]{a} s_2 \Leftrightarrow_{\mathrm{Def}} s_2 \in \delta(s_1, a).$$

Wir sagen dann, $a$ *schaltet* oder *feuert* von $s_1$ nach $s_2$. Sind $s' \in S$ und $a \in A$, so daß $\delta(s', a) \neq \varnothing$ gilt, so heißt $a$ *schaltfähig* oder *konzessioniert* bei $s'$. In diesem Fall schreiben wir $s' \xrightarrow[\mathcal{T}]{a}$. Die Menge

$$\mathrm{en}_{\mathcal{T}}(s') =_{\mathrm{Def}} \left\{ a \in A : s' \xrightarrow[\mathcal{T}]{a} \right\}$$

bezeichnet die Menge der bei einem Zustand $s' \in S$ konzessionierten Elemente aus $A$.

$\mathcal{T}$ ist *endlich*, wenn $S$ endlich ist. □

Ist $A$ ein Alphabet zu einem Transititionssystem, so bezeichnen wir die Elemente aus $A$ als *Aktionen*.

I.3.2. Beispiel. Abbildung I.8 zeigt ein endliches Transitionssystem. Kreise repräsentieren Zustände, Pfeile werden zur Darstellung der Transitionsrelation verwendet. Der Initialzustand wird durch eine einlaufende Pfeilspitze gekennzeichnet. □

I.3.3. Definition. Ein Transitionssystem $\mathcal{T}$ über $A$ heißt *deterministisch*, wenn für alle $s, s_1, s_2 \in S_{\mathcal{T}}$ und für alle $a \in A$

$$s \xrightarrow[\mathcal{T}]{a} s_1 \;\&\; s \xrightarrow[\mathcal{T}]{a} s_2 \Rightarrow s_1 = s_2$$

gilt. Ist $\mathcal{T}$ ein deterministisches Transitionssystem über $A$, so fassen wir die Transitionsfunktion $\delta_{\mathcal{T}}$ als partielle Abbildung $\delta_{\mathcal{T}} : S_{\mathcal{T}} \times A \rightharpoonup S_{\mathcal{T}}$ auf. □

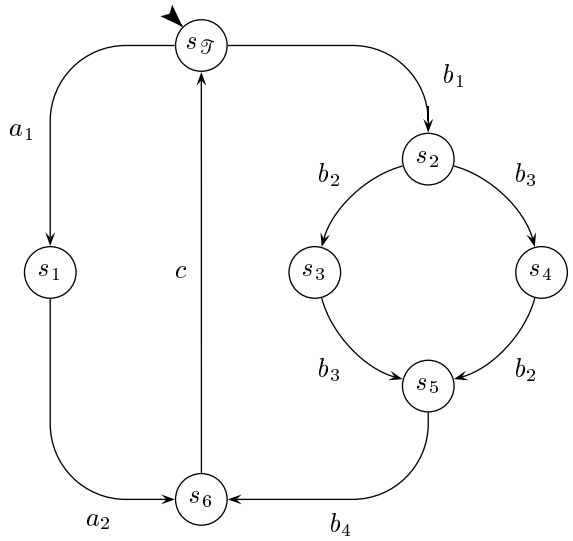



ABBILDUNG I.8. Ein Transitionssystem

I.3.4. BEMERKUNG. Zur Klärung des Begriffs „Nichtdeterminismus“: In der Literatur taucht dieser Begriff in zwei verschiedenen und leicht zu verwechselnden Bedeutungen auf: (1) Nichtdeterministische Auswahl zwischen verschiedenen, unterscheidbaren Verhaltensalternativen und (2) Auswahl von nichtunterscheidbaren Alternativen.

Betrachten wir hierzu prozeßalgebraische Terme (mit Petri-Netzen kann der folgende Sachverhalt nicht dargestellt werden) der Form $\tau_1 = a.(b+c)$ und $\tau_2 = a.(b+b)$. $a, b$, und $c$ sind Systemaktionen, . bezeichnet Hintereinanderausführung und + die nichtdeterministische Auswahl. $\tau_1$ ist also so zu lesen, daß zunächst $a$ und dann entweder $b$ oder $c$ ausgeführt werden. $\tau_2$ bedeutet: Nach Ausführung von $a$ findet eine Auswahl zwischen $b$ und $b$ statt (was sinnvoll ist, wenn $b$ zur eine Abstraktion von konkreteren Aktionen darstellt). $\tau_1$ ist nichtdeterministisch im Sinne von (1), jedoch deterministisch im Sinne von (2), während $\tau_2$ in beiden Bedeutungen nichtdeterministisch ist. Diese Arbeit verwendet die Bedeutung (1). □

I.3.5. BEISPIEL (Netzsystem). Ein *Netz* $\langle P, T, F\rangle$ besteht aus einer endlichen Menge $P$ von *Plätzen*, einer endlichen Menge $T$ von *Transitionen* und einer *Flußrelation* $F \subseteq (P \times T) \cup (T \times P)$, wobei $P \cap T = \varnothing$ gilt. Wir nehmen an, daß für alle Transitionen $t \in T$ ein Platz $p \in P$ mit $pFt$ oder $tFp$ existiert. Ein *Zustand* (auch als *Markierung* oder *Fall* bezeichnet) eines Netzes ist eine Menge $Q \subseteq P$. Ist $\langle P, T, F\rangle$ ein Netz und $Q$ einer seiner Zustände, so nennen wir die Struktur $\mathcal{N} = \langle P, T, F, Q\rangle$ ein *Netzsystem*. $Q$ wird als *initialer Zustand* von $\mathcal{N}$ bezeichnet.

Abbildung I.9 zeigt ein Netzsystem. Plätze werden als Kreise dargestellt, Transitionen als Rechtecke oder Quadrate. Pfeile stellen die Flußrelation dar. Elemente des initialen

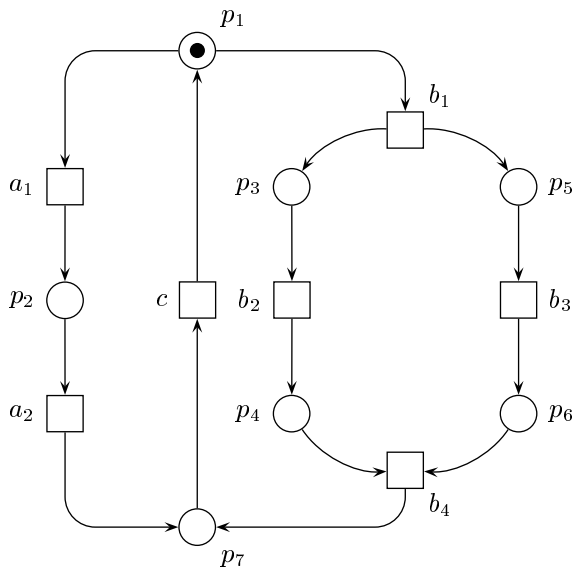



ABBILDUNG I.9. Ein Netzsystem.

Zustands werden durch einen schwarzen Punkt gekennzeichnet, den wir als *Marke* bezeichnen. Ist ein Platz $p$ in einer Markierung $Q$ enthalten, so nennen wir $p$ bei $Q$ *markiert*; wir sagen auch, der Platz $p$ *trägt* eine Marke.

Wie gewöhnlich verwenden wir die Notationen

$$\dot{}x = F_{\mathcal{N}}^{-1}(x), x^{\dot{}} = F_{\mathcal{N}}(x) \text{ und } \dot{x} = \dot{}x \cup x^{\dot{}} \text{ bzw.}$$
$$\dot{}X = F_{\mathcal{N}}^{-1}(X), X^{\dot{}} = F_{\mathcal{N}}(X) \text{ und } \dot{X} = \dot{}X \cup X^{\dot{}}$$

für $x \in P_{\mathcal{N}} \cup T_{\mathcal{N}}$ und $X \subseteq P_{\mathcal{N}} \cup T_{\mathcal{N}}$.

Eine Transition $t \in T_{\mathcal{N}}$ heißt *schaltfähig* oder *konzessioniert* bei einem Zustand $Q \subseteq P_{\mathcal{N}}$, wenn $\dot{}t \subseteq Q$ und $(t^{\dot{}} - \dot{}t) \cap Q = \varnothing$ gilt, wir scheiben dann $Q \xrightarrow[\mathcal{N}]{t}$.

I.3.6. BEMERKUNG. Die in dieser Arbeit verwendete Klasse von Petri-Netzen sind — obwohl die Definition einer Markierung als Menge von Plätzen dies zu implizieren scheint — keine elementaren Netzsysteme [**60, 75**]. Der Unterschied besteht in der Schaltregel: Für elementare Netzsystem gilt die folgende Konzessioniertheitsbedingung: $\dot{}t \subseteq Q$ und $t^{\dot{}} \cap Q = \varnothing$ für einen Zustand $Q \subseteq P_{\mathcal{N}}$ und eine Transition $t \in T_{\mathcal{N}}$. □

Die *Transitionsfunktion* $\delta_{\mathcal{N}} : \mathscr{P}(P_{\mathcal{N}}) \times T_{\mathcal{N}} \rightharpoonup \mathscr{P}(P_{\mathcal{N}})$ eine Netzsystems ist als

$$\delta_{\mathcal{N}}(Q,t) =_{\text{Def}} \begin{cases} (Q - \dot{}t) \cup t^{\dot{}}, \text{falls } Q \xrightarrow[\mathcal{N}]{t}; \\ \text{undefiniert sonst.} \end{cases}$$

definiert. Weiterhin sei $S \subseteq \mathscr{P}(P_{\mathcal{N}})$ die kleinste Menge mit

$$Q_{\mathcal{N}} \in S \text{ und } \bigcup_{t \in T_{\mathcal{N}}} \delta_{\mathcal{N}}(S,t) \subseteq S.$$

Die Struktur $\mathcal{T}(\mathcal{N}) =_{\text{Def}} \langle S, \delta_{\mathcal{N}} \restriction S \times T_{\mathcal{N}}, Q_{\mathcal{N}} \rangle$ heißt das *durch $\mathcal{N}$ induzierte Transitionssystem*. Offenbar handelt es sich bei $\mathcal{T}(\mathcal{N})$ um ein deterministisches Transitionssystem.

Das Transitionssystem aus Abbildung I.8 wird zu dem durch das Netzsystem aus Abbildung I.9 induzierte Transitionssystem, wenn wir seine Zustände wie folgt wählen:

$$s_{\mathcal{T}} = \{p_1\}, s_1 = \{p_2\}, s_2 = \{p_3, p_5\}, s_3 = \{p_4, p_5\}$$
$$s_4 = \{p_3, p_6\}, s_5 = \{p_4, p_6\}, s_6 = \{p_7\}.$$

Ein Netzsystem $\mathcal{N}$ heißt *sicher* oder *kontaktfrei*, wenn

$$\dot{t} \subseteq Q \Rightarrow (t^{\cdot} - \dot{t}) \cap Q = \varnothing$$

für alle $t \in T_{\mathcal{N}}$ und für alle $Q \in S_{\mathcal{T}(\mathcal{N})}$ gilt. Das Netzsystem aus Abb. I.9 ist kontaktfrei. □

I.3.7. Definition (Sprache eines Transitionssystems). Ist $\mathcal{T}$ ein Transitionssystem über $A$, so erweitern wir seine Transitionsfunktion auf Sequenzen über $A$, indem wir

$$\delta^*_{\mathcal{T}}(s, \epsilon) =_{\text{Def}} \{s\}, \text{und } \delta^*_{\mathcal{T}}(s, \sigma a) =_{\text{Def}} \delta_{\mathcal{T}}(\delta^*_{\mathcal{T}}(s, \sigma), a)$$

für $\sigma \in A^*$ und $a \in A$ setzen. Ist $\mathcal{T}$ deterministisch, so gilt offenbar $|\delta^*_{\mathcal{T}}(s, \sigma)| \leqslant 1$ für alle $s \in S_{\mathcal{T}}$ und $\sigma \in A^*$; in diesem Fall fassen wir deshalb $\delta^*_{\mathcal{T}}$ als partielle Abbildung $S_{\mathcal{T}} \times A^* \rightharpoonup S_{\mathcal{T}}$ auf.

Ist $s \in S_{\mathcal{T}}$ und $\sigma \in A^*$, schreiben wir $s \overset{\sigma}{\Longrightarrow}_{\mathcal{T}}$, falls $\delta_{\mathcal{T}}(s, \sigma) \neq \varnothing$ bzw. im deterministischen Fall definiert ist. Wiederum verwenden wir die „Pfeilnotation“:

$$s \overset{\sigma}{\Longrightarrow}_{\mathcal{T}} s' \Leftrightarrow_{\text{Def}} s' \in \delta^*_{\mathcal{T}}(s, \sigma).$$

Die *Sprache* von $\mathcal{T}$ ist

$$\mathbf{L}(\mathcal{T}) =_{\text{Def}} \left\{ \sigma \in A^* : s_{\mathcal{T}} \overset{\sigma}{\Longrightarrow}_{\mathcal{T}} \right\}.$$

Elemente aus $\mathbf{L}(\mathcal{T})$ werden auch als *Schaltfolgen* von $\mathcal{T}$ bezeichnet. □

I.3.8. Bemerkung. Um die Sprechweisen zu vereinfachen, nennen wir eine Sequenz $\sigma$ von Aktionen eine Schaltfolge eines Netzsystems $\mathcal{N}$, wenn $\sigma$ Schaltfolge des durch $\mathcal{N}$ induzierten Transitionssystem ist. Analoge Konventionen gelten für die weiter unten eingeführten Spur- und Semisprachen. □

I.3.9. Definition. Sind $\mathcal{T}_1$ und $\mathcal{T}_2$ Transitionssysteme über demselben Alphabet $A$, so nennen wir $\mathcal{T}_1$ und $\mathcal{T}_2$ *isomorph*, wenn es eine Bijektion $f : S_{\mathcal{T}_1} \to S_{\mathcal{T}_2}$ gibt, so daß $f(s_{\mathcal{T}_1}) = s_{\mathcal{T}_2}$ und

$$s_1 \xrightarrow[\mathcal{T}_1]{a} s_2 \Leftrightarrow f(s_1) \xrightarrow[\mathcal{T}_2]{a} f(s_2)$$

gilt. □

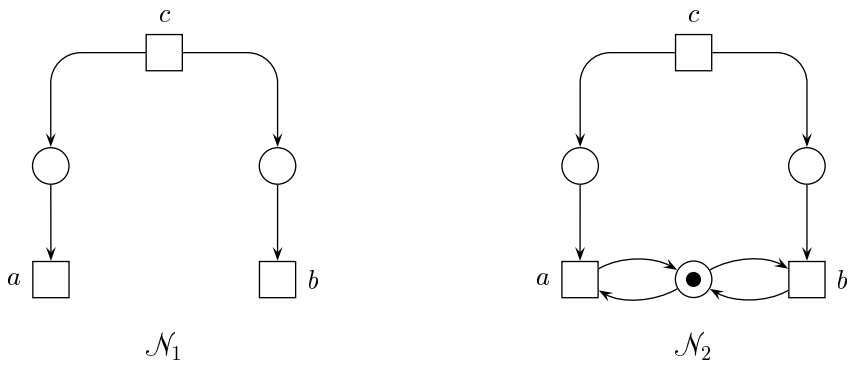



ABBILDUNG I.10. Zwei Netzsysteme mit isomorphen Transitionssystemen

**Spursysteme.** Fassen wir die notwendigen Bedingungen zur Definition einer Unabhängigkeitsbeziehung $I$ zusammen, die Paare von Systemaktionen charakterisiert, die nebenläufig zueinander stattfinden können:

(a) Keine Systemaktion kann unabhängig von sich selbst stattfinden, d. h. $I$ ist irreflexiv, und
(b) findet $a$ unabhängig von $b$ statt, so ist $b$ unabhängig von $a$, d. h. $I$ ist symmetrisch.

Wenn wir Transitionssysteme um einen expliziten Begriff von Nebenläufigkeit erweitern wollen, müssen wir strukturelle Forderungen an ein solches System stellen, die sicherstellen, daß unabhängige Aktionen tatsächlich in beliebiger Reihenfolge auftreten können. Diese Eigenschaften (Definition I.3.12, (a) bis (c)) werden aus naheliegenden Gründen als *Rauteneigenschaften* bezeichnet.

I.3.10. BEMERKUNG. Die Aussagen „Die Systemaktionen $a$ und $b$ finden nebenläufig statt“ und „Die Systemaktionen $a$ und $b$ finden in beliebiger Reihenfolge statt“ können nicht gleichgesetzt werden. Zwar impliziert der erste Satz den zweiten, die Umkehrung gilt jedoch nicht. In dem in Abbildung I.10 dargestellen Netzsystem $\mathcal{N}_1$ etwa sind $a$ und $b$ unabhängig, in $\mathcal{N}_2$ hingegen nicht (s. Beispiel I.3.15 für eine formale Definition der Unabhängigkeitsrelation für Netzsysteme). Dennoch sind die induzierten Transitonssysteme beider Netzsysteme isomorph (vgl. Abb. I.11). □

I.3.11. DEFINITION (Spuralphabet). Ein *Spuralphabet* $\Sigma = \langle A, I \rangle$ besteht aus einem Alphabet $A$ von *Aktionen* und einer symmetrischen und irreflexiven Relation $I \subseteq A \times A$, die als *Unabhängigkeitsrelation* bezeichnet wird. Ihr Komplement $D =_{\text{Def}} \bar{I}$ heißt *Abhängigkeitsrelation.* □

Ist $\Sigma$ ein Spuralphabet, so bezeichnen wir die zugehörige Abhängigkeitsrelation mit $D_\Sigma$. Offenbar ist $D_\Sigma$ eine symmetrische und reflexive Relation.

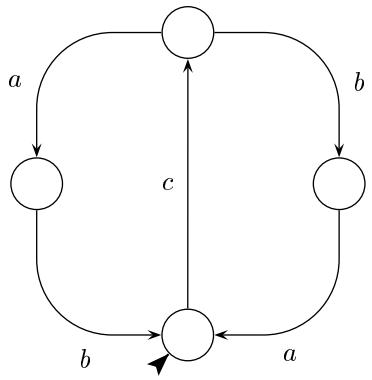



ABBILDUNG I.11. Induziertes Transitionssystem der Netze aus Abb. I.10

I.3.12. DEFINITION (Spursystem). Sei $\Sigma$ ein Spuralphabet und sei $\mathcal{T}$ ein deterministisches Transitionssystem über $A_\Sigma$. $\mathcal{T}$ heißt *Spursystem über* $\Sigma$, wenn für alle $s_1, s_2, s_3 \in S_\mathcal{T}$ und $a, b \in A_\Sigma$ mit $a\ I_\Sigma\ b$ die folgenden Bedingungen erfüllt sind:

(a) $s_1 \xrightarrow[\mathcal{T}]{a} s_2 \ \&\ s_1 \xrightarrow[\mathcal{T}]{b} s_3 \Rightarrow \exists s_4 \in S_\mathcal{T} \left( s_2 \xrightarrow[\mathcal{T}]{b} s_4 \ \&\ s_3 \xrightarrow[\mathcal{T}]{a} s_4 \right)$,

(b) $s_1 \xrightarrow[\mathcal{T}]{a} s_2 \ \&\ s_3 \xrightarrow[\mathcal{T}]{b} s_2 \Rightarrow \exists s_4 \in S_\mathcal{T} \left( s_4 \xrightarrow[\mathcal{T}]{b} s_1 \ \&\ s_4 \xrightarrow[\mathcal{T}]{a} s_3 \right)$, sowie

(c) $s_1 \xrightarrow[\mathcal{T}]{a} s_2 \xrightarrow[\mathcal{T}]{b} s_3 \Rightarrow \exists s_4 \in S_\mathcal{T} \left( s_1 \xrightarrow[\mathcal{T}]{b} s_4 \xrightarrow[\mathcal{T}]{a} s_3 \right)$.

□

Abbildung I.12 illustriert die Bedingungen (a), (b) und (c) aus Definition I.3.12. Wir zählen einige einfache Folgerungen aus Definition I.3.12 auf:

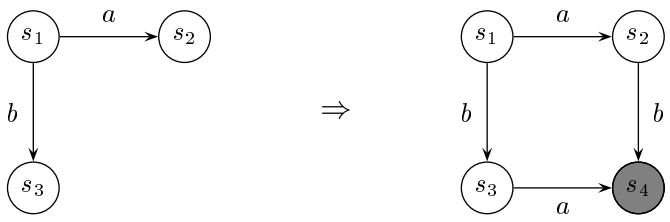
I.3.13. LEMMA. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$ und seien $a_1, a_2 \in A_\Sigma$ mit $a_1\ I_\Sigma\ a_2$. Seien $s_1, s_2 \in S_\mathcal{T}$.*

(a) $\neg s_1 \xrightarrow[\mathcal{T}]{a_1} \ \&\ s_1 \xrightarrow[\mathcal{T}]{a_2} s_2 \Rightarrow \neg s_2 \xrightarrow[\mathcal{T}]{a_1}$.

(b) $\neg s_2 \xrightarrow[\mathcal{T}]{a_2} \ \&\ s_1 \xrightarrow[\mathcal{T}]{a_1} s_2 \Rightarrow \neg s_1 \xrightarrow[\mathcal{T}]{a_2}$.

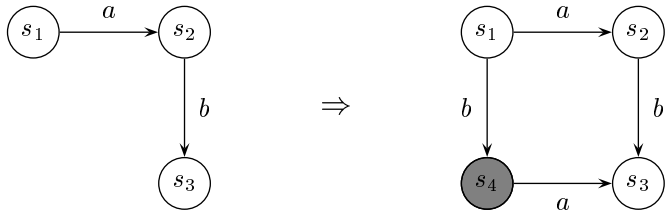
BEWEIS. (a) Nehmen wir $\neg s_1 \xrightarrow[\mathcal{T}]{a_1} \ \&\ s_1 \xrightarrow[\mathcal{T}]{a_2} s_2 \xrightarrow[\mathcal{T}]{a_1} s_3$ für ein $s_3 \in S_\mathcal{T}$ an. Aus $s_1 \xrightarrow[\mathcal{T}]{a_2} s_2 \xrightarrow[\mathcal{T}]{a_1} s_3$ folgt jedoch nach Definition I.3.12.(c), daß es ein $s_4 \in S_\mathcal{T}$ mit $s_1 \xrightarrow[\mathcal{T}]{a_1} s_4 \xrightarrow[\mathcal{T}]{a_2} s_3$ gibt im Widerspruch zu $\neg s_1 \xrightarrow[\mathcal{T}]{a_1}$.

(b) $s_1 \xrightarrow[\mathcal{T}]{a_1} s_2 \ \&\ s_1 \xrightarrow[\mathcal{T}]{a_2} s_3$ für ein $s_3 \in S_\mathcal{T}$ impliziert nach Definition I.3.12.(a) die Existenz eines Zustands $s_4 \in S_\mathcal{T}$ mit $s_3 \xrightarrow[\mathcal{T}]{a_1} s_4$ und insbesondere $s_2 \xrightarrow[\mathcal{T}]{a_2} s_4$. □




Bedingung (a)


Bedingung (b)




Bedingung (c)

ABBILDUNG I.12. Illustration von Definition I.3.12

---

I.3.14. LEMMA. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$. Sei $a \in A_\Sigma$ eine Aktion und $\sigma \in A_\Sigma^*$ eine Aktionssequenz mit $\sigma = a_0 a_1 \dots a_{n-1}$, so daß $a\ I_\Sigma\ a_i$ für $0 \leqslant i < n$ mit $n \geqslant 0$ und $s \xRightarrow[\mathcal{T}]{\sigma} s'$ für Zustände $s, s' \in S_\mathcal{T}$ gilt. Dann gilt $s \xrightarrow[\mathcal{T}]{a}$ gdw. $s' \xrightarrow[\mathcal{T}]{a}$. Weiterhin folgt $\delta^*_\mathcal{T}(s, \sigma a) = \delta^*_\mathcal{T}(s, a\sigma)$ aus $s \xRightarrow[\mathcal{T}]{\sigma a}$.*

BEWEIS. Induktion über $n = |\sigma|$. Für $n = 0$ gilt $s = s'$ und nichts ist zu zeigen. Nehmen wir $n > 0$ sowie als Induktionsvoraussetzung

$$s \xrightarrow[\mathcal{T}]{a} \Leftrightarrow s' \xrightarrow[\mathcal{T}]{a} \text{ und } s \xRightarrow[\mathcal{T}]{\sigma a} \Rightarrow \delta^*_\mathcal{T}(s, \sigma a) = \delta^*_\mathcal{T}(s, a\sigma)$$

---

an. Sei $a_n \in A_\Sigma$ mit $s' \xrightarrow[\mathcal{T}]{a_n} s''$ für einen Zustand $s'' \in S_\mathcal{T}$, so daß $a\ I_\Sigma\ a_n$ vorliegt. Gilt $s \xrightarrow[\mathcal{T}]{a}$, so auch $s' \xrightarrow[\mathcal{T}]{a}$ nach Induktionsvoraussetzung. Mit Definition I.3.12.(a) folgt nun $s'' \xrightarrow[\mathcal{T}]{a}$. Eine weitere Anwendung von Definition I.3.12.(a) und der zweite Teil der Induktionsvoraussetzung liefern uns

$$s \xRightarrow[\mathcal{T}]{\sigma a_n a} \quad \Rightarrow \quad \delta^*_\mathcal{T}(s, \sigma a_n a) = \delta^*_\mathcal{T}(s, a\sigma a_n).$$

Nehmen wir andernfalls $\neg s \xrightarrow[\mathcal{T}]{a}$ an, erhalten wir mit der Induktionsvoraussetzung $\neg s' \xrightarrow[\mathcal{T}]{a}$, und mit Hilfe von Lemma I.3.13.(a) $\neg s'' \xrightarrow[\mathcal{T}]{a}$. □

I.3.15. Beispiel (Netzsysteme). Für ein Netzsystem $\mathcal{N}$ definieren wir die Relation $I_\mathcal{N} \subseteq T_\mathcal{N} \times T_\mathcal{N}$ als

$$t_1\ I_\mathcal{N}\ t_2 \Leftrightarrow_{\text{Def}} \dot{t}_1 \cap \dot{t}_2 = \varnothing. \tag{I.3.$\alpha$}$$

Betrachten wir das Netzsystem aus Abbildung I.9, so können wir die folgende Tabelle aufstellen:

| | $a_1$ | $a_2$ | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $c$ |
|---|---|---|---|---|---|---|---|
| $a_1$ | $D$ | $D$ | $D$ | $I$ | $I$ | $I$ | $I$ |
| $a_2$ | $D$ | $D$ | $I$ | $I$ | $I$ | $D$ | $D$ |
| $b_1$ | $D$ | $I$ | $D$ | $D$ | $D$ | $I$ | $I$ |
| $b_2$ | $I$ | $I$ | $D$ | $D$ | $I$ | $D$ | $I$ |
| $b_3$ | $I$ | $I$ | $D$ | $I$ | $D$ | $D$ | $I$ |
| $b_4$ | $I$ | $D$ | $I$ | $D$ | $D$ | $D$ | $D$ |
| $c$ | $I$ | $D$ | $I$ | $I$ | $I$ | $D$ | $D$ |

Es ist leicht nachzuweisen, daß die Struktur $\Sigma_\mathcal{N} =_{\text{Def}} \langle T_\mathcal{N}, I_\mathcal{N} \rangle$ ein Spuralphabet ist. Weiterhin ist das durch $\mathcal{N}$ induzierte Transitonssystem $\mathcal{T}(\mathcal{N})$ ein Spursystem. Wir weisen exemplarisch Punkt (a) aus Definition I.3.12 nach. Seien $t_1, t_2 \in T_\mathcal{N}$ mit $t_1\ I_\mathcal{N}\ t_2$ und $Q_1$, $Q_2$ und $Q_3$ Zustände von $\mathcal{N}$. Nehmen wir $Q_1 \xrightarrow[\mathcal{N}]{t_1} Q_2$ und $Q_1 \xrightarrow[\mathcal{N}]{t_2} Q_3$ an, d. h.

$$Q_2 = (Q_1 - \dot{t}_1) \cup t_1^\bullet \text{ und } Q_3 = (Q_1 - \dot{t}_2) \cup t_2^\bullet \tag{I.3.$\beta$}$$

Da jedoch wegen Festlegung (I.3.$\alpha$) $\dot{t}_1 \cap \dot{t}_2 = \varnothing$ gilt, folgt

$$\dot{t}_2 \subseteq Q_2\ \&\ (t_2^\bullet - \dot{t}_2) \cap Q_2 = \varnothing \text{ und } \dot{t}_1 \subseteq Q_3\ \&\ (t_1^\bullet - \dot{t}_1) \cap Q_3 = \varnothing$$

d. h. es gibt Zustände $Q_4$ und $Q_4'$, so daß $Q_2 \xrightarrow[\mathcal{N}]{t_2} Q_4$ und $Q_3 \xrightarrow[\mathcal{N}]{t_1} Q_4'$ gilt. Es verbleibt $Q_4 = Q_4'$ zu zeigen:

$$\begin{aligned} Q_4 &= (Q_2 - {}^\cdot t_2) \cup t_2^\cdot \\ &= (((Q_1 - {}^\cdot t_1) \cup t_1^\cdot) - {}^\cdot t_2) \cup t_2^\cdot && \text{(nach (I.3.}\beta\text{))} \\ &= (((Q_1 - {}^\cdot t_2) \cup t_2^\cdot) - {}^\cdot t_1) \cup t_1^\cdot && \text{(nach (I.3.}\alpha\text{))} \\ &= (Q_3 - {}^\cdot t_1) \cup t_1^\cdot && \text{(nach (I.3.}\beta\text{))} \\ &= Q_4' \end{aligned}$$

□

**Halbordnungssemantik I: Spuren.** Definition I.3.12 beschreibt, in welcher Weise unabhängige Ereignisse in einem Spursystem $\mathcal{T}$ über $\Sigma$ auftreten können. Ein externer Beobachter von $\mathcal{T}$ kann unabhängige Ereignisse in beliebiger Reihenfolge wahrnehmen: Ist $\sigma ab\rho$ eine Schaltfolge von $\mathcal{T}$ ($\sigma, \rho \in A_\Sigma^*$, $a, b \in A_\Sigma$ mit $a\ I_\Sigma\ b$), kann ebenso die Schaltfolge $\sigma ba\rho$ beobachtet werden, beide Schaltfolgen sind also äquivalent bzgl. ihrer Beobachtbarkeit.

I.3.16. Definition (Mazurkiewiczspur). Sei $\Sigma$ ein Spuralphabet. Wir definieren eine Relation $\sim_\Sigma \subseteq A_\Sigma^* \times A_\Sigma^*$ durch

$$\sigma \sim_\Sigma \rho \Leftrightarrow_{\text{Def}} \exists \sigma_1, \sigma_2 \in A_\Sigma^*, a, b \in A_\Sigma\, (a\ I_\Sigma\ b\ \&\ \sigma = \sigma_1 ab\sigma_2\ \&\ \rho = \sigma_1 ba\sigma_2)\,.$$

Weiterhin setzen wir $\approx_\Sigma =_{\text{Def}} \sim_\Sigma^*$. Anders gesagt, $\sigma \approx_\Sigma \rho$ gilt genau dann, wenn $\rho$ aus $\sigma$ durch eine endliche Folge von Vertauschung adjazenter unabhägiger Zeichen konstruiert werden kann.

Offenbar ist $\approx_\Sigma$ eine Kongruenz über dem Monoid $\mathcal{M}(A_\Sigma) =_{\text{Def}} \langle A_\Sigma^*, \cdot, \epsilon\rangle$. Die Äquivalenzklasse einer Sequenz $\sigma \in A_\Sigma^*$ bzgl. $\approx_\Sigma$ notieren wir mit

$$[\sigma]_\Sigma =_{\text{Def}} \{\rho \in A_\Sigma^* : \sigma \approx_\Sigma \rho\}$$

und nennen sie eine *Mazurkiewiczspur* oder kurz eine *Spur* über $\Sigma$. $\mathbf{TR}(\Sigma)$ bezeichnet die Klasse der Spuren über $\Sigma$. □

I.3.17. Beispiel. Sei $\Sigma$ ein Spuralphabet mit den Komponenten $A_\Sigma = \{a, b, c, d\}$ und der $I_\Sigma = \{\langle a, c\rangle, \langle c, a\rangle, \langle b, c\rangle, \langle c, b\rangle\}$. Abbildung I.13 zeigt ein Netzsystem $\mathcal{N}$ mit einer entsprechenden Unabhängigkeitsrelation. *dacdbc* ist eine Schaltfolge von $\mathcal{N}$; die zugehörige Spur ist

$$[dacdbc]_\Sigma = \{dacdbc, dcadbc, dacdcb, dcadcb\}.$$

□

I.3.18. Bemerkung. In der Literatur wird $\approx_\Sigma$ häufig wie folgt eingeführt: Sind $a, b \in A_\Sigma$ Symbole eines Spuralphabets $\Sigma$, so wird die Relation $\sim_\Sigma' \subseteq A_\Sigma^* \times A_\Sigma^*$ als

$$ab \sim_\Sigma' ba \Leftrightarrow_{\text{Def}} a\ I_\Sigma\ b$$

definiert. $\approx_\Sigma$ ist dann die kleinste Kongruenz über dem Monoid $\langle A_\Sigma^*, \cdot, \epsilon\rangle$, die $\sim_\Sigma'$ enthält.

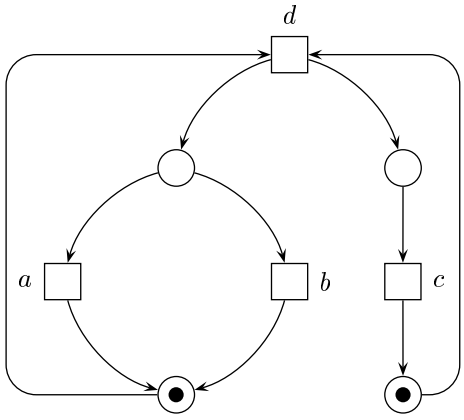



ABBILDUNG I.13. Ein Netzsystem zur Illustration des Spurbegriffs.

Aus der allgemeinen Algebra ist bekannt, daß die Struktur $\mathscr{M}(\Sigma) = \langle \mathbf{TR}(\Sigma), \cdot, [\epsilon]_\Sigma \rangle$ mit

$$[\sigma]_\Sigma \cdot [\rho]_\Sigma =_{\text{Def}} [\sigma \cdot \rho]_\Sigma$$

ebenfalls ein Monoid ist — wir nennen dieses Monoid das *Spurmonoid* über $\Sigma$. □

I.3.19. LEMMA. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$. Für alle $\sigma, \rho \in A_\Sigma^*$ mit $\sigma \approx_\Sigma \rho$ und für alle $s_1, s_2 \in S_\mathcal{T}$ gilt*

$$s_1 \overset{\sigma}{\underset{\mathcal{T}}{\Longrightarrow}} s_2 \quad \Leftrightarrow \quad s_1 \overset{\rho}{\underset{\mathcal{T}}{\Longrightarrow}} s_2 .$$

BEWEIS. Eine einfache Induktion über der Folge von Vertauschungen adjazenter unabhängiger Symbole, die von $\sigma$ zu $\rho$ führt. Dabei wird Def. I.3.12.(c) im Induktionsschritt angewendet. □

I.3.20. DEFINITION (Spursprache). Die *Spursprache* eines Spursystems $\mathcal{T}$ über $\Sigma$ ist

$$\mathbf{TRL}(\mathcal{T}) =_{\text{Def}} \{[\sigma]_\Sigma \in \mathbf{TR}(\Sigma) : \sigma \in \mathbf{L}(\mathcal{T})\} .$$

□

**Halbordnungssemantik II: Semisprachen.** Wir haben Spuren als Semantik nebenläufiger Systeme eingeführt. Nebenläufigkeit bleibt im Spurbegriff jedoch hinter der Vertauschbarkeit unabhängiger adjazenter Symbole verborgen — die Rekonstruktion der Unabhängigkeitsrelation aus den Elementen einer Spur ist im Einzelfall keine Aufgabe, die durch bloßes Hinsehen gelöst werden kann.

Auch im Hinblick auf die Diskussion in der Einleitung, die Halbordnungen als geeignete mathematische Struktur für Kausalbeziehungen identifiziert, stellt sich

die Frage, ob nicht etwa eine Semantik für nebenläufige Systeme direkt auf der Grundlage beschrifteter Halbordnungen definiert werden kann.

Wir beginnen dieses Unterfangen mit einer Verschärfung des Begriffs der Semiordnung: Wir definieren solche Semiordnungen, deren Unabhängigkeitsrelation mit der Unabhängigkeitsrelation eines zugrundeliegenden Spuralphabets übereinstimmt.

I.3.21. Definition. Für ein SpuralphabetSpuralphabet $\Sigma$ enthalte die Klasse $\mathbf{CSO}(\Sigma)$ diejenigen Semiordnungen $x$ über $A_\Sigma$, für die

$$e_1 \operatorname{co}_x e_2 \Rightarrow \lambda_x(e_1)\, I_\Sigma\, \lambda_x(e_2)$$

für alle $e_1, e_2 \in E_x$ gilt; Elemente dieser Klasse heißen *$\Sigma$-konsistent*.

$$\mathbf{CSW}(\Sigma) =_{\text{Def}} \{\boldsymbol{x} \in \mathbf{SW}(A_\Sigma) : x \in \mathbf{CSO}(\Sigma)\}$$

ist die zugehörige Klasse *$\Sigma$-konsistenter* Semiwörter. □

Da wir im folgenden weniger mit Definition I.3.21 als mit ihrer Kontraposition arbeiten werden, ist es nützlich, diese explizit aufzuführen:

I.3.22. Lemma. *Ist $\boldsymbol{x} \in \mathbf{CSW}(\Sigma)$, so folgt für alle $e_1, e_2 \in E_x$ aus $\lambda_x(e_1)\, D_\Sigma\, \lambda_x(e_2)$ bereits $e_1 \operatorname{li}_x e_2$.*

Für die Buchstaben eines Aktionsalphabets $A$ und auch für Zeichenketten über einem solchen Alphabet haben wir Konzessioniertheitsbedingungen und Transitionsfunktionen bestimmt. Nun diskutieren wir die Konzessioniertheit von Semiordnungen.

I.3.23. Definition (Schaltfunktion für Semiordnungen). Ist $\mathcal{T}$ ein Spursystem über $\Sigma$, so definieren wir für alle $x \in \mathbf{SO}(A_\Sigma)$ eine Schaltfunktion für Semiordnungen $\delta^x_{\mathcal{T}} : S_{\mathcal{T}} \times \mathbf{C}(x) \to \mathcal{P}(S_{\mathcal{T}})$. Für $s \in S_{\mathcal{T}}$ und $C \in \mathbf{C}(x)$ sei

$$\delta^x_{\mathcal{T}}(s, C) =_{\text{Def}} \{s' \in S_{\mathcal{T}} : \exists \sigma \in \operatorname{lin}_x(C)\, (s' = \delta^*_{\mathcal{T}}(s, \sigma) \text{ definiert})\}\,. \qquad \text{(I.3.}\gamma\text{)}$$

Weiterhin setzen wir $\delta_{\mathcal{T}}(s, x) =_{\text{Def}} \delta^x_{\mathcal{T}}(s, E_x)$. □

Vor allem anderen können wir feststellen:

I.3.24. Lemma. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$ und seien $x, y \in \mathbf{SO}(A_\Sigma)$. Für alle $s \in S_{\mathcal{T}}$ gilt $x \equiv y \Rightarrow \delta_{\mathcal{T}}(s, x) = \delta_{\mathcal{T}}(s, y)$.*

Beweis. Direkt mit Hilfe von Lemma I.2.9.(b). □

Die Berechnung einer Zustandsmenge nach Festlegung (I.3.$\gamma$) muß für jede mögliche Auswahl einer Linearisierung von $C$ erfolgen. Natürlich liefern i. Allg. unterschiedliche Auswahlen von Linearisierungen $\sigma$ verschiedene Folgezustände $s' = \delta^*_{\mathcal{T}}(s, \sigma)$. Betrachten wir jedoch $\Sigma$-konsistente Semiordnungen (deren unabhängige Elemente ja mit unabhängigen Aktionen beschriftet sind), ist es möglich nachzuweisen, daß das Ergebnis der Auswertung von $\delta^*_{\mathcal{T}}(s, \sigma)$ unabhängig von der konkreten Wahl von $\sigma$ ist. Wir benötigen dazu das folgende Lemma:

I.3.25. LEMMA. *Sei $\Sigma$ ein Spuralphabet und sei $x \in \mathbf{CSO}(\Sigma)$. Dann gilt $\sigma \approx_\Sigma \rho$ für alle $\sigma, \rho \in \mathrm{lin}(x)$.*

BEWEIS. $\rho$ ist offensichtlich eine Permutation von $\sigma$, d. h. $\rho$ kann aus $\sigma$ durch eine endliche Folge von Vertauschungen adjezenter Symbole konstruiert werden. Da jedoch $x \in \mathbf{CSO}(\Sigma)$ gilt, folgt aus $\lambda_x(e_1)\ D_\Sigma\ \lambda_x(e_2)$ und $e_1 <_x e_2$ sowohl $G_x^\sigma(e_1) <_\sigma G_x^\sigma(e_2)$ als auch $G_x^\rho(e_1) <_\rho G_x^\rho(e_2)$. Damit ist es nur notwendig, adjazente unabhängige Symbole zur Konstruktion von $\rho$ aus $\sigma$ zu vertauschen. □

I.3.26. LEMMA. *Ist $x \in \mathbf{CSO}(\Sigma)$ und ist $\mathcal{T}$ ein Spursystem über $\Sigma$, $C \in \mathbf{C}(x)$ und $s \in S_\mathcal{T}$, so gilt*

(a) $\delta^*_\mathcal{T}(s, \sigma) = \delta^*_\mathcal{T}(s, \rho)$ *für alle* $\sigma, \rho \in \mathrm{lin}_x(C)$,
(b) $|\delta^x_\mathcal{T}(s, C)| \leqslant 1$ *für alle* $s \in S_\mathcal{T}$.

BEWEIS. (a) folgt aus den Lemmata I.3.19 und I.3.25.
(b) folgt aus (a) und dem Umstand, daß $\delta^*_\mathcal{T}$ eine partielle Funktion ist. □

Aufgrund dieses Lemmas fassen wir von nun an $\delta^x_\mathcal{T}$ als partielle Abbildung $\delta^x_\mathcal{T} : S_\mathcal{T} \times \mathbf{C}(x) \rightharpoonup S_\mathcal{T}$ auf, wenn $\mathcal{T}$ ein Spursystem über $\Sigma$ und $x \in \mathbf{CSO}(\Sigma)$ ist.

I.3.27. DEFINITION. Sei $x \in \mathbf{CSO}(\Sigma)$ und $\mathcal{T}$ ein Spursystem über $\Sigma$. $x$ heißt *schaltfähig* oder *konzessioniert* bei einem Zustand $s \in S_\mathcal{T}$, wenn für jedes $e \in E_x$

$$\delta^x_\mathcal{T}\left(s, <_x^{-1}(e)\right) \xrightarrow[\mathcal{T}]{\lambda_x(e)}$$

gilt. Wir schreiben in diesem Fall $s \overset{x}{\Longrightarrow}_\mathcal{T}$. Unter Verweis auf Lemma I.3.24 legen wir die *Semisprache* von $\mathcal{T}$ als

$$\mathbf{SL}(\mathcal{T}) =_{\mathrm{Def}} \left\{ \boldsymbol{x} \in \mathbf{CSW}(\Sigma) : s_\mathcal{T} \overset{x}{\Longrightarrow}_\mathcal{T} \right\}.$$

fest. Weiterhin verwenden wir noch

$$s_1 \overset{x}{\Longrightarrow}_\mathcal{T} s_2 \Leftrightarrow_{\mathrm{Def}} s_2 = \delta_\mathcal{T}(s_1, x) \text{ definiert.}$$

□

I.3.28. BEISPIEL. Die in Abbildung I.1 gezeigten Semiordnungen sind bei dem Initialzustand $Q_\mathcal{N}$ des Netzsystems aus Abbildung I.9 schaltfähig. □

Unser Kriterium für die Schaltfähigkeit von Semiordnungen $x$ weicht von den Schaltkriterien für Semiordnungen in Petri-Netzen ab, die in [**23**], [**78**] und [**95**] angegeben sind. Vogler [**95**] etwa definiert Schaltfähigkeit für Semiordnungen $x$ wie folgt: Jede unabhängige Menge $C$ von $x$ ist schaltfähig, nachdem alle Ereignisse einer beliebigen Menge $B \in \mathbf{C}(x)$ mit $<_x^{-1}(C) \subseteq B$ und $B \cap C = \varnothing$ stattgefunden haben. Dabei kann für allgemeine Petri-Netze eine Definition für die Schaltfähigkeit einer (Multi)-Menge von Aktionen angegeben werden, die nicht auf dem Begriff der Sequenz beruht. In Starkes Version [**78**] der Definition von Schaltfähigkeit ist $B$ maximal gewählt (d. h. $B = E_x - \leqslant_x(C)$), während in [**23**] $B$ minimal ist, also

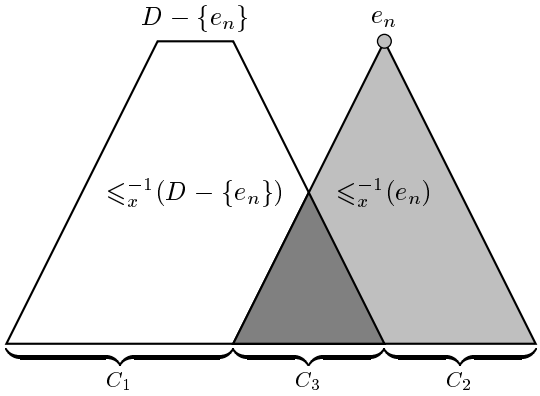



ABBILDUNG I.14. Illustration zum Beweis von Lemma I.3.29

$B = <_x^{-1}(C)$ gilt. Es ist leicht nachzuweisen, daß alle drei Definitionen äquivalent sind. Wir zeigen nun, daß unsere Definition äquivalent zu der in [**23**] angegebenen ist.

I.3.29. LEMMA. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$, $x \in \mathbf{CSO}(\Sigma)$ und $s \in S_{\mathcal{T}}$. Dann gilt*

$$s \xRightarrow[\mathcal{T}]{x} \;\Leftrightarrow\; \forall C \in \mathbf{C}(x) \left[ \forall e \in \max_{\leqslant_x}(C) \left( \delta^x_{\mathcal{T}}(s, C - \{e\}) \xrightarrow[\mathcal{T}]{\lambda_x(e)} \right) \right]$$

BEWEIS. ($\Leftarrow$) ist offensichtlich, da $\leqslant_x^{-1}(e) \in \mathbf{C}(x)$ und $\max_{\leqslant_x}(\leqslant_x^{-1}(e)) = \{e\}$ ist.

($\Rightarrow$) Sei $D = \{e_1, e_2, \ldots, e_n\} = \max_{\leqslant_x}(C)$ für eine Menge $C \in \mathbf{C}(x)$. Der Beweis wird mit Hilfe einer Induktion über der Größe $|D|$ unabhängiger Mengen $D$ erbracht. Für $n = 1$ ist $C = \leqslant_x^{-1}(e_1)$. Nach Voraussetzung gilt aber $\delta^x_{\mathcal{T}}(s, \leqslant_x^{-1}(e_1)) \xrightarrow[\mathcal{T}]{\lambda_x(e_1)}$.

Sei $n > 1$. Partitionieren wir die Menge $C = \leqslant_x^{-1}(D)$ in der folgenden Weise:

$$C_1 = \leqslant_x^{-1}(D - \{e_n\}) - \leqslant_x^{-1}(e_n), C_2 = \leqslant_x^{-1}(e_n) - \leqslant_x^{-1}(D - \{e_n\}) \text{ und}$$
$$C_3 = \leqslant_x^{-1}(D - \{e_n\}) \cap \leqslant_x^{-1}(e_n).$$

Abbildung I.14 illustriert diese Partitionierung. Nach Induktionsvoraussetzung ist $\delta^x_{\mathcal{T}}(s, C_1 \cup C_3)$ definiert, und da $x$ bei $s$ schaltfähig ist, ist $\delta^x_{\mathcal{T}}(s, C_2 \cup C_3)$ ebenfalls definiert. Es folgt, daß es Sequenzen $\sigma_i \in \mathrm{lin}_x(C_i)$ für $i = 1, 2, 3$ gibt, so daß $\delta^*_{\mathcal{T}}(s, \sigma_3\sigma_1)$ und ebenso $\delta^*_{\mathcal{T}}(s, \sigma_3\sigma_2)$ definiert ist. Schließlich beobachten wir noch, daß $\lambda_x^{-1}(C_1) \times \lambda_x^{-1}(C_2) \subseteq I_\Sigma$ gilt. Das Lemma folgt nun mit Hilfe einer einfachen

Induktion über der Länge von $\sigma_2$, wobei von Lemma I.3.14 Gebrauch gemacht wird. □

Die beiden folgenden Theoreme verallgemeinern entsprechende Theoreme von Starke [**78**]. Wir zeigen, daß Semisprachen von Spursystemen abgeschlossen bzgl. Präfixbildung und Sequentialisierung sind.

I.3.30. THEOREM. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$, $x, y \in \mathbf{CSO}(\Sigma)$ und sei $s \in S_{\mathcal{T}}$. Dann gilt $s \xRightarrow[\mathcal{T}]{x}$ & $y \leqslant x \Rightarrow s \xRightarrow[\mathcal{T}]{y}$.*

BEWEIS. Sei $e \in E_y$. Da $H_y^x$ eine Einbettung ist, gilt

$$y\left[\leqslant_y^{-1}(e)\right] \;\equiv\; x\left[\leqslant_x^{-1}\left(H_y^x(e)\right)\right].$$

Wir rechnen

$$\begin{aligned}
&y\left[\leqslant_y^{-1}(e)\right] \equiv x\left[\leqslant_x^{-1}\left(H_y^x(e)\right)\right] \\
&\Rightarrow y\left[<_y^{-1}(e)\right] \equiv x\left[<_x^{-1}\left(H_y^x(e)\right)\right] && (H_y^x \text{ Einbettung}) \\
&\Rightarrow \mathrm{lin}\left(y\left[<_y^{-1}(e)\right]\right) = \mathrm{lin}\left(x\left[<_x^{-1}\left(H_y^x(e)\right)\right]\right) && (\text{Lemma I.2.9.(b)}) \\
&\Rightarrow \mathrm{lin}_y\left(<_y^{-1}(e)\right) = \mathrm{lin}_x\left(<_x^{-1}\left(H_x^y(e)\right)\right) && (\forall z \in \mathbf{SO}(A_\Sigma)(\mathrm{lin}(z) = \mathrm{lin}_z(E_z))) \\
&\Rightarrow \delta_{\mathcal{T}}^y\left(s, <_y^{-1}(e)\right) = \delta_{\mathcal{T}}^x\left(s, <_x^{-1}\left(H_y^x(e)\right)\right) && (\text{Def. I.3.23}) \\
&\Rightarrow \left(\delta_{\mathcal{T}}^y\left(s, <_y^{-1}(e)\right) \xrightarrow[\mathcal{T}]{\lambda_y(e)} \Leftrightarrow \delta_{\mathcal{T}}^x\left(s, <_x^{-1}\left(H_y^x(e)\right)\right) \xrightarrow[\mathcal{T}]{\lambda_x(H_y^x(e))}\right). \\
& && (\lambda_y(e) = \lambda_x(H_y^x(e)))
\end{aligned}$$

Mit der Voraussetzung $s \xRightarrow[\mathcal{T}]{x}$ folgt nun das Theorem. □

I.3.31. THEOREM. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$, $x, y \in \mathbf{CSO}(\Sigma)$ und $s_1, s_2 \in S_{\mathcal{T}}$. Dann gilt $s_1 \xRightarrow[\mathcal{T}]{x} s_2$ & $x \preccurlyeq y \Rightarrow s_1 \xRightarrow[\mathcal{T}]{y} s_2$.*

BEWEIS. Sei $e \in E_x$. Da $x$ nach Voraussetzung schaltfähig bei $s_1$ ist, gilt

$$\delta_{\mathcal{T}}^*(s_1, \sigma) \xrightarrow[\mathcal{T}]{\lambda_x(e)}$$

unter Anwendung von Lemma I.3.26.(a) für alle $\sigma \in \mathrm{lin}_x\left(<_x^{-1}(e)\right)$. Setzen wir

$$B = \leqslant_x^{-1}(e), C_1 = \leqslant_x^{-1}\left(G_x^y(e)\right) \text{ und } C_2 = G_x^y\left(\leqslant_x^{-1}(e)\right)$$

Es gilt $C_1 \subseteq C_2$ und weiterhin

$$(\lambda_y(C_2 - C_1) \times \lambda_y(C_1)) \cup (\lambda_y(C_1) \times \lambda_y(C_2 - C_1)) \subseteq I_\Sigma, \qquad (\text{I.3.}\delta)$$

da $x, y \in \mathbf{CSO}(\Sigma)$ sind. Betrachten wir nun eine Linearisierung $\rho$ von $C_2$. $\rho$ ist von der Form

$$\rho = \rho_0 \lambda_y(G_x^y(e_0))\rho_1 \lambda_y(G_x^y(e_1))\rho_2 \ldots \rho_{n-1}\lambda_y(G_x^y(e_{n-1}))\rho_n \lambda_y(G_x^y(e_n))$$

wobei $\sigma =_{\text{Def}} \lambda_x(e_0)\lambda_x(e_1)\ldots\lambda_x(e_{n-1})\lambda_x(e_n)$ eine Linearisierung von $B$ mit $e = e_n$ ist. Sei $\rho_j = \lambda_y(e_{j0})\lambda_y(e_{j1})\ldots\lambda_y(e_{jm_j-1})$ für ein $m_j \geqslant 0$. Mit (I.3.$\delta$) folgt $\lambda_y(e_{jk})\ I_\Sigma\ \lambda_x(e_i)$ für $0 \leqslant i < n$, $0 \leqslant j \leqslant n$ und $0 \leqslant k < m_j$.

Wir zeigen nun mit Hilfe einer Induktion über $n$, daß für alle $n \geqslant 0$

$$\delta^*_{\mathcal{J}}(s_1, \lambda_x(e_0)\lambda_x(e_1)\ldots\lambda_x(e_{n-1})) \xrightarrow[\mathcal{J}]{\lambda_x(e_n)}$$
$$\Rightarrow\ \delta^*_{\mathcal{J}}(s_1, \rho_0\lambda_y(G^y_x(e_0))\rho_1\lambda_y(G^y_x(e_1))\rho_2\ldots\rho_{n-1}\lambda_y(G^y_x(e_{n-1}))\rho_n) \xrightarrow[\mathcal{J}]{\lambda_y(G^y_x(e_n))}$$

gilt. Für $n = 0$ haben wir die Implikation

$$s_1 \xrightarrow[\mathcal{J}]{\lambda_x(e_0)} \ \Rightarrow\ \delta^*_{\mathcal{J}}(s_1, \rho_0) \xrightarrow[\mathcal{J}]{\lambda_y(G^y_x(e_0))}$$

nachzuweisen. Falls $\delta^*_{\mathcal{J}}(S_1, \rho_0)$ definiert ist, können wir Lemma I.3.14 anwenden: Da $\lambda_y(G^y_x(e_0))\ I_\Sigma\ \lambda_y(e_{0j})$ nach (I.3.$\delta$) für $0 \leqslant j < m_0$ gilt, berechnen wir

$$s_1 \xrightarrow[\mathcal{J}]{\lambda_x(e_0)} \Rightarrow s_1 \xrightarrow[\mathcal{J}]{\lambda_y(G^y_x(e_0))} \qquad (\lambda_x(e_0) = \lambda_y(G^y_x(e_0)))$$
$$\Rightarrow \delta^*_{\mathcal{J}}(s_1, \rho_0) \xrightarrow[\mathcal{J}]{\lambda_y(G^y_x(e_0))} \qquad \text{(Lemma I.3.14)}$$

Nehmen wir also $\neg s_1 \overset{\rho_0}{\underset{\mathcal{J}}{\Longrightarrow}}$ an. Sei $k < m_0$ der kleinste Index, so daß ein Zustand $s \in S_{\mathcal{J}}$ mit

$$\delta^*_{\mathcal{J}}(s_1, \lambda_y(e_{00})\lambda_y(e_{01})\ldots\lambda_y(e_{0k-1})) = s \text{ und } \neg s \xrightarrow[\mathcal{J}]{e_{0k}}$$

existiert. Setzen wir nun

$$\begin{aligned} \hat{e}_{0j} &=_{\text{Def}} G^{y-1}_x(e_{0j}) \text{ für } 0 \leqslant j \leqslant k, \\ \tau &=_{\text{Def}} \lambda_x(\hat{e}_{00})\lambda_x(\hat{e}_{01})\ldots\lambda_x(\hat{e}_{0k-1}) \text{ und} \\ C &=_{\text{Def}} \{\hat{e}_{00}, \hat{e}_{01}, \ldots, \hat{e}_{0k-1}\}, \end{aligned}$$

so ist $\tau \in \text{lin}_x(C)$. Offenbar gilt $C \in \mathbf{C}(x)$ ebenso wie $C \cup \{e_k\} \in \mathbf{C}(x)$. Es folgt $\neg\delta_{\mathcal{J}}(s_1, \tau) \xrightarrow[\mathcal{J}]{\lambda_x(\hat{e}_k)}$. Da $\tau$ eine Linearisierung von $C$ ist, können wir $\neg\delta^x_{\mathcal{J}}(s_1, C) \xrightarrow[\mathcal{J}]{\lambda_x(\hat{e}_k)}$ folgern. Damit gilt jedoch nach Lemma I.3.29 $\neg s_1 \overset{x}{\underset{\mathcal{J}}{\Longrightarrow}}$. Wir schließen $s_1 \overset{\rho_0}{\underset{\mathcal{J}}{\Longrightarrow}}$.

Sei $n > 0$. Unsere Induktionshypothese ist

$$\delta^*_{\mathcal{J}}(s_1, \lambda_x(e_0)\lambda_x(e_1)\ldots\lambda_x(e_{n-2})) \xrightarrow[\mathcal{J}]{\lambda_x(e_{n-1})}$$
$$\Rightarrow\ \delta^*_{\mathcal{J}}(s_1, \rho_0\lambda_y(G^y_x(e_0))\rho_1\lambda_y(G^y_x(e_1))\rho_2$$
$$\ldots\rho_{n-2}\lambda_y(G^y_x(e_{n-2}))\rho_{n-1}) \xrightarrow[\mathcal{J}]{\lambda_y(G^y_x(e_{n-1}))}.$$

Offenbar erfolgt der Induktionsschritt nun mit einer Argumentation, die analog zu der für den Fall $n = 0$ erfolgen kann: Ist $\rho_n$ bei $\delta^*_{\mathcal{J}}(s_1, \lambda_x(e_0)\lambda_x(e_1)\ldots\lambda_x(e_{n-1}))$ konzessioniert, können wir Lemma I.3.14 anwenden, andernfalls betrachten wir die längste schaltfähige Folge, die ein Präfix von $\rho_n$ ist. Das Urbild dieser Folge unter

$G_x^y$ ist nun ebenfalls nicht schaltfähig. Da es sich dabei aber um die Linearisierung einer Menge $C \in \mathbf{C}(x)$ handelt, können wir mit Hilfe von Lemma I.3.29 folgern, daß $x$ bei $s_{\mathcal{T}}$ nicht konzessioniert ist.

Das Theorem folgt dann mit der Beobachtung, daß $\mathrm{lin}(x) \supseteq \mathrm{lin}(y)$ gilt, d. h. $\delta_{\mathcal{T}}(s_1, x) = s_2 = \delta_{\mathcal{T}}(s_1, y)$. □

I.3.32. Korrolar. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$ und sei $\boldsymbol{x} \in \mathbf{SL}(\mathcal{T})$. Dann gilt $\boldsymbol{y} \leqslant \boldsymbol{x} \vee \boldsymbol{x} \preccurlyeq \boldsymbol{y} \Rightarrow \boldsymbol{y} \in \mathbf{SL}(\mathcal{T})$ für alle $\boldsymbol{y} \in \mathbf{SW}(A_\Sigma)$.*

I.3.33. Korrolar. *Sind $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{CSW}(\Sigma)$, ist $\mathcal{T}$ ein Spursystem über $\Sigma$ und sind $s_1, s_2, s_3 \in S_{\mathcal{T}}$ Zustände, so gilt*

$$s_1 \xRightarrow[\mathcal{T}]{x} s_2 \;\&\; s_2 \xRightarrow[\mathcal{T}]{y} s_3 \Rightarrow s_1 \xRightarrow[\mathcal{T}]{xy} s_3$$

Beweis. Das Theorem folgt sofort aus der Beobachtung, daß es für alle $\sigma \in \mathrm{lin}(xy)$ Linearisierungen $\rho \in \mathrm{lin}(x)$ und $\tau \in \mathrm{lin}(y)$ gibt, so daß $\sigma = \rho\tau$ ist. □

**Semisprachen kleinster Sequentialität.** Wir wenden unsere Aufmerksamkeit nun der Ordnung $\preccurlyeq$ auf Semiordnungen bzw. Semiwörtern zu. Wir haben bereits festgestellt, daß Semisprachen von Spursystemen abgeschlossen bzgl. $\preccurlyeq$ sind. Eine naheliegende Frage ist, unter welchen Umständen für ein Spursystem $\mathcal{T}$ eine minimale Sprache $X \subseteq \mathbf{SL}(\mathcal{T})$ gefunden werden kann, so daß die Menge der Sequentialiserungen von Elementen aus $X$ gerade $\mathbf{SL}(\mathcal{T})$ ergibt. Wir werden zeigen, daß eine derartige Sprache $X$ für ein deterministisches Spursystem existiert und eindeutig konstruktiv bestimmt werden kann. Wir definieren weiterhin eine Operation $\circ_\Sigma$, die es erlaubt, derartige Semiordnungen bzw. Semiwörter kleinster Sequentialität miteinander zu verketten. Diese Operation wird in Abschnitt III.1 zur Definition eines adäqaten Sprachbegriffs für Prozeßautomaten verwendet werden.

I.3.34. Definition. Wir definieren die Operation $\langle\cdot\rangle_\Sigma : \mathbf{SO}(\Sigma) \to \mathbf{SO}(\Sigma)$ als $\langle x\rangle_\Sigma =_{\mathrm{Def}} \langle E_x, <_{\langle x\rangle_\Sigma}, \lambda_x\rangle$ mit

$$e_1 <_{\langle x\rangle_\Sigma} e_2 \Leftrightarrow_{\mathrm{Def}} e_1 <_x e_1 \;\&\; \lambda_x(e_1)\; D_\Sigma^+\; \lambda_x(e_2)$$

□

I.3.35. Lemma. *Seien $x, y \in \mathbf{SO}(\Sigma)$.*

(a) $\langle x\rangle_\Sigma \in \mathbf{SO}(\Sigma)$,
(b) $x \in \mathbf{CSO}(\Sigma) \Rightarrow \langle x\rangle_\Sigma \in \mathbf{CSO}(\Sigma)$,
(c) $x \leqslant y \Rightarrow \langle x\rangle_\Sigma \leqslant \langle y\rangle_\Sigma$,
(d) $x \in \mathbf{CSO}(\Sigma) \;\&\; x \preccurlyeq y \Rightarrow \langle x\rangle_\Sigma \equiv \langle y\rangle_\Sigma$.

Beweis. (a) $<_{\langle x\rangle_\Sigma}$ ist eine irreflexive Halbordung, da $<_x$ eine Halbordnung und $D_\Sigma^+$ antisymmetrisch ist. Sind $e_1, e_2 \in E_{\langle x\rangle_\Sigma}$ mit $e_1 \neq e_2$ und $\lambda_{\langle x\rangle_\Sigma}(e_1) = \lambda_{\langle x\rangle_\Sigma}(e_2)$, gilt — da $x$ eine Semiordnung ist — einerseits $e_1\ \mathrm{li}_x\ e_2$. Andererseits ist jedoch $D_\Sigma^+$ reflexiv (da $D_\Sigma$ reflexiv ist), es gilt also $\lambda_x(e_1)\; D_\Sigma^+\; \lambda_x(e_2)$. Es folgt $e_1\ \mathrm{li}_{\langle x\rangle_\Sigma}\ e_2$.

(b) folgt direkt aus Lemma I.3.22.
(c) Wir berechnen

$$\begin{aligned} & e_1 \in <_x^{-1}(e_2) \;\&\; \lambda_x(e_1)\; D_\Sigma^+\; \lambda_x(e_1) \\ & \quad \Leftrightarrow \;\; H_x^y(e_1) \in <_x^{-1}(H_x^y(e_2)) \;\&\; \lambda_x(H_x^y(e_1))\; D_\Sigma^+\; \lambda_x(H_x^y(e_1)) \end{aligned}$$

für alle $e_1, e_2 \in E_x$.

(d) $\langle x\rangle_\Sigma \preccurlyeq \langle y\rangle_\Sigma$ ist offensichtlich, nehmen wir also $\langle y\rangle_\Sigma \not\preccurlyeq \langle x\rangle_\Sigma$ an. Dann gibt es Ereignisse $e_1, e_2 \in E_{\langle y\rangle_\Sigma}$, so daß $e_1 \operatorname{co}_{\langle y\rangle_\Sigma} e_2$, jedoch entweder $G_y^x(e_1) <_{\langle x\rangle_\Sigma} G_y^x(e_2)$ oder $G_y^x(e_2) <_{\langle x\rangle_\Sigma} G_y^x(e_1)$ gilt. Im ersten Fall impliziert $e_1 \operatorname{co}_{\langle y\rangle_\Sigma} e_2$ jedoch $\neg\lambda_y(e_1)\; D_\Sigma^+\; \lambda_y(e_2)$, woraus wiederum $G_y^x(e_1) \operatorname{co}_{\langle x\rangle_\Sigma} G_y^x(e_2)$ folgt. Der zweite Fall ist analog. □

I.3.36. Definition. Mit Lemma I.3.35.(c) können wir $\langle\cdot\rangle_\Sigma$ mit Hilfe der Definition

$$\langle \boldsymbol{x}\rangle_\Sigma =_{\text{Def}} [\langle x\rangle_\Sigma]$$

zu einer Operation $\mathbf{SW}(\Sigma) \to \mathbf{SW}(\Sigma)$ erweitern. □

I.3.37. Theorem. *Seien $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{SL}(\mathcal{T})$ für ein Spursystem $\mathcal{T}$ über $\Sigma$. Wenn es eine Semiordnung $\boldsymbol{z} \in \mathbf{CSO}(\Sigma))$ mit $\boldsymbol{z} \preccurlyeq \boldsymbol{x}$ und $\boldsymbol{z} \preccurlyeq \boldsymbol{y}$ gibt, so folgt*

(a) $\langle \boldsymbol{x}\rangle_\Sigma = \langle \boldsymbol{y}\rangle_\Sigma$,
(b) $\langle \boldsymbol{x}\rangle_\Sigma \preccurlyeq \boldsymbol{z}$,
(c) $\langle \boldsymbol{x}\rangle_\Sigma \in \mathbf{SL}(\mathcal{T})$.

Beweis. (a) $\langle x\rangle_\Sigma \equiv \langle z\rangle_\Sigma \equiv \langle y\rangle_\Sigma$ gilt nach Lemma I.3.35.(d).
(b) Wir beobachten $\langle x\rangle_\Sigma \equiv \langle z\rangle_\Sigma \preccurlyeq z$ ebenfalls mit Hilfe von Lemma I.3.35.(d).
(c) Nehmen wir $\langle \boldsymbol{x}\rangle_\Sigma \notin \mathbf{SL}(\mathcal{T})$ an; es gibt also ein $e \in E_{\langle x\rangle_\Sigma}$, so daß

$$\neg\delta_{\mathcal{T}}^{\langle x\rangle_\Sigma}\left(s_{\mathcal{T}}, <_x^{-1}(e)\right) \xrightarrow[\mathcal{T}]{\lambda_{\langle x\rangle_\Sigma}(e)}$$

gilt. Da $\langle x\rangle_\Sigma \preccurlyeq x$ gilt, können wir die Menge $C =_{\text{Def}} G_{\langle x\rangle_\Sigma}^x(<_x^{-1}(e))$ betrachten: Es gilt $C \in \mathbf{C}(x)$. Da nun jedoch wegen Lemma I.2.9.(a)

$$\operatorname{lin}_{\langle x\rangle_\Sigma}\left(<_x^{-1}(e)\right) \supseteq \operatorname{lin}_x(C)$$

gilt, folgt

$$\neg\delta_{\mathcal{T}}^x\left(s_{\mathcal{T}}, C\right) \xrightarrow[\mathcal{T}]{\lambda_{\langle x\rangle_\Sigma}\left(G_{\langle x\rangle_\Sigma}^x(e)\right)} .$$

Mit Lemma I.3.29 folgern wir $\boldsymbol{x} \notin \mathbf{SL}(\mathcal{T})$. □

Die Operation $\langle\cdot\rangle_\Sigma$ konstruiert also für ein Semiwort $\boldsymbol{x} \in \mathbf{SL}(\mathcal{T})$ dasjenige Semiwort $\langle \boldsymbol{x}\rangle_\Sigma$, das minimal bzgl. $\preccurlyeq$ und noch in $\mathbf{SL}(\mathcal{T})$ enthalten ist. Die Operation $\circ_\Sigma$, deren Definition wir nun angeben, erlaubt die Konkatenation zweier Semiwörter $\boldsymbol{x}$ und $\boldsymbol{y}$ in der Weise, daß $\langle \boldsymbol{x}\rangle_\Sigma \circ_\Sigma \langle \boldsymbol{y}\rangle_\Sigma = \langle \boldsymbol{xy}\rangle_\Sigma$ gilt (vgl. Theorem I.3.41.(d)).

I.3.38. DEFINITION (Schwach-sequentielle Kompositon). Die Operation $\circ_\Sigma : \mathbf{SO}(A_\Sigma) \times \mathbf{SO}(A_\Sigma) \to \mathbf{SO}(A_\Sigma)$, die als *schwach-sequentielle Komposition* bezeichnet wird, ist als

$$x \circ_\Sigma y =_{\text{Def}} \left\langle E_x \cup E_y, (\leqslant_x \cup \leqslant_y \cup R)^+, \lambda_x \cup \lambda_y \right\rangle$$

definiert, wobei $E_x \cap E_y = \varnothing$ vorausgesetzt und $R \subseteq E_x \times E_y$ durch

$$e_1 \; R \; e_2 \Leftrightarrow_{\text{Def}} \lambda_x(e_1) \; D_\Sigma \; \lambda_y(e_2)$$

gegeben ist. □

I.3.39. LEMMA. *Seien $x_1, x_2, y_1, y_2 \in \mathbf{SO}(A_\Sigma)$ für ein Spuralphabet $\Sigma$. Dann gilt*

$$x_1 \equiv x_2 \;\&\; y_1 \equiv y_2 \Rightarrow x_1 \circ_\Sigma y_1 \equiv x_2 \circ_\Sigma y_2.$$

BEWEIS. Es gilt $x_1 \equiv x_2$ gdw. $\gamma(x_1) = \gamma(x_2)$ nach Theorem I.2.29.(e), analog für $y_1$ und $y_2$. Da $\circ_\Sigma$ offenbar wohldefiniert ist, folgt

$$\gamma(x_1) \circ_\Sigma \gamma(y_1) = \gamma(x_2) \circ_\Sigma \gamma(y_2) \quad \Rightarrow \quad \gamma(x_1 \circ_\Sigma y_1) = \gamma(x_2 \circ_\Sigma y_2)$$

Eine weitere Anwendung von Theorem I.2.29.(e) liefert uns

$$x_1 \circ_\Sigma y_1 \equiv x_2 \circ_\Sigma y_2$$

□

I.3.40. DEFINITION. Mit Lemma I.3.39 definieren wir

$$\boldsymbol{x} \circ_\Sigma \boldsymbol{y} =_{\text{Def}} [x \circ_\Sigma y]$$

für alle Semiwörter $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{SW}(A_\Sigma)$ über der Alphabetkomponente eines Spuralphabets $\Sigma$. □

I.3.41. THEOREM. *Für alle $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{x} \in \mathbf{CSW}(\Sigma)$ gelten die folgenden Äquivalenzen:*

(a) $\boldsymbol{x} \circ_\Sigma \boldsymbol{\epsilon} = \boldsymbol{\epsilon} \circ_\Sigma \boldsymbol{x} = \boldsymbol{x}$
(b) $(\boldsymbol{x} \circ_\Sigma \boldsymbol{y}) \circ_\Sigma \boldsymbol{z} = \boldsymbol{x} \circ_\Sigma (\boldsymbol{y} \circ_\Sigma \boldsymbol{z})$,
(c) $\langle \boldsymbol{\epsilon} \rangle_\Sigma = \boldsymbol{\epsilon}$,
(d) $\langle \boldsymbol{xy} \rangle_\Sigma = \langle \boldsymbol{x} \rangle_\Sigma \circ_\Sigma \langle \boldsymbol{y} \rangle_\Sigma$,
(e) $\langle \boldsymbol{x} \circ_\Sigma \boldsymbol{y} \rangle_\Sigma = \langle \boldsymbol{x} \rangle_\Sigma \circ_\Sigma \langle \boldsymbol{y} \rangle_\Sigma$.

BEWEIS. (a) bis (c) sind offensichtlich.

(d) Offenbar gilt $E_{\langle xy \rangle_\Sigma} = E_{\langle x \rangle_\Sigma \circ_\Sigma \langle y \rangle_\Sigma}$ und $\lambda_{\langle xy \rangle_\Sigma} = \lambda_{\langle x \rangle_\Sigma \circ_\Sigma \langle y \rangle_\Sigma}$, es bleibt also nur $<_{\langle xy \rangle_\Sigma} = <_{\langle x \rangle_\Sigma \circ_\Sigma \langle y \rangle_\Sigma}$ zu zeigen. Seien $e_1, e_2 \in E_{\langle xy \rangle_\Sigma}$ mit $e_1 <_{\langle xy \rangle_\Sigma} e_2$. Es gilt

$$\begin{aligned}
&\big(e_1 \in E_x \;\&\; e_2 \in E_x \;\&\; e_1 <_x e_2 \;\&\; \lambda_x(e_1) \; D_\Sigma^+ \; \lambda_x(e_2)\big) && \text{(Fall 1)} \\
\vee\; &\big(e_1 \in E_y \;\&\; e_2 \in E_y \;\&\; e_1 <_y e_2 \;\&\; \lambda_y(e_1) \; D_\Sigma^+ \; \lambda_y(e_2)\big) && \text{(Fall 2)} \\
\vee\; &\big(e_1 \in E_x \;\&\; e_2 \in E_y \;\&\; \lambda_x(e_1) \; D_\Sigma^+ \; \lambda_y(e_2)\big) && \text{(Fall 3)}
\end{aligned}$$

Fall 1. Es folgt $e_1 <_{\langle x \rangle_\Sigma} e_2$. Mit $<_{\langle x \rangle_\Sigma} \subseteq <_{\langle x \rangle_\Sigma \circ_\Sigma \langle y \rangle_\Sigma}$ gilt dann $e_1 <_{\langle x \rangle_\Sigma \circ_\Sigma \langle y \rangle_\Sigma} e_2$.

Fall 2. Analog zu Fall 1.
Fall 3. $e_1 <_{\langle x\rangle_\Sigma \circ_\Sigma \langle y\rangle_\Sigma} e_2$ folgt mit Definition I.3.38.
(e) Analog zu (d) □

I.3.42. DEFINITION. Sei $\Sigma$ ein Spuralphabet. Wir setzen

$$\mathbf{LSO}(\Sigma) =_{\text{Def}} \{\langle x\rangle_\Sigma \in \mathbf{SO}(\Sigma) : x \in \mathbf{SO}(\Sigma)\},$$
$$\mathbf{LSW}(\Sigma) =_{\text{Def}} \{\boldsymbol{x} \in \mathbf{SW}(\Sigma) : x \in \mathbf{LSO}(\Sigma)\}.$$

Ist $\mathcal{T}$ ein Spursystem über $\Sigma$, so definieren wir weiterhin

$$\mathbf{LSSL}(\mathcal{T}) =_{\text{Def}} \{\langle \boldsymbol{x}\rangle_\Sigma \in \mathbf{CSW}(\Sigma) : \boldsymbol{x} \in \mathbf{SL}(\mathcal{T})\}.$$

□

I.3.43. KORROLAR. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$, $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{CSW}(\Sigma)$ und $s_1, s_2, s_3 \in S_{\mathcal{T}}$. Dann gilt*

$$s_1 \xRightarrow[\mathcal{T}]{\langle x\rangle_\Sigma} s_2 \;\&\; s_2 \xRightarrow[\mathcal{T}]{\langle y\rangle_\Sigma} s_3 \Rightarrow s_1 \xRightarrow[\mathcal{T}]{\langle x\rangle_\Sigma \circ_\Sigma \langle y\rangle_\Sigma} s_3.$$

BEWEIS. Wir rechnen

$$\begin{aligned}
s_1 \xRightarrow[\mathcal{T}]{\langle x\rangle_\Sigma} s_2 \;\&\; s_2 \xRightarrow[\mathcal{T}]{\langle y\rangle_\Sigma} s_3 &\Rightarrow s_1 \xRightarrow[\mathcal{T}]{\langle x\rangle_\Sigma \langle y\rangle_\Sigma} s_3 && \text{(Korollar I.3.33)}\\
&\Rightarrow s_1 \xRightarrow[\mathcal{T}]{\left\langle \langle x\rangle_\Sigma \langle y\rangle_\Sigma \right\rangle_\Sigma} s_3 && \text{(Theorem I.3.37.(c))}\\
&\Rightarrow s_1 \xRightarrow[\mathcal{T}]{\langle x\rangle_\Sigma \circ_\Sigma \langle y\rangle_\Sigma} s_3. && \text{(Theorem I.3.41.(d))}
\end{aligned}$$

□

**Äquivalenz von Spur- und Semisprachen.** Wir charakterisieren das Verhältnis von Spur- und Semisprachen, indem wir den algebraischen Begriff des Homomorphismus verwenden. Im Allgemeinen bezeichnen wir algebraische Strukturen $\mathcal{X} = \langle A_{\mathcal{X}}, f^1_{\mathcal{X}}, f^2_{\mathcal{X}}, \ldots, f^k_{\mathcal{X}}\rangle$ und $\mathcal{Y} = \langle A_{\mathcal{Y}}, f^1_{\mathcal{Y}}, f^2_{\mathcal{Y}}, \ldots, f^k_{\mathcal{Y}}\rangle$ als *vergleichbar*, wenn die Operationen $f^i_{\mathcal{X}} : A_{\mathcal{X}} \times \cdots \times A_{\mathcal{X}} \to A_{\mathcal{X}}$ und $f^i_{\mathcal{Y}} : A_{\mathcal{Y}} \times \cdots \times A_{\mathcal{Y}} \to A_{\mathcal{X}}$ $(1 \leqslant i \leqslant k)$ dieselbe Anzahl von Parametern haben, wobei wir Konstanten als 0-stellige Operationen auffassen. Vergleichbare Strukturen $\mathcal{X}$ und $\mathcal{Y}$ heißen *homomorph*, wenn es eine Abbildung $h : A_{\mathcal{X}} \to A_{\mathcal{Y}}$ gibt, die die Gleichung

$$f^i_{\mathcal{Y}}(h(a_1), h(a_2), \ldots, h(a_n)) = h\left(f^i_{\mathcal{X}}(a_1, a_2, \ldots, a_n)\right)$$

für $1 \leqslant i \leqslant k$ erfüllt; $h$ wird dann als *Homomorphismus* bezeichnet. Ist $h$ bijektiv, so nennen wir $h$ einen *Isomorphismus* und die Strukturen $\mathcal{X}$ und $\mathcal{Y}$ *isomorph*.

I.3.44. DEFINITION. Wir definieren die Struktur $\mathcal{S}(\Sigma) = \langle \mathbf{LSO}(\Sigma), \circ_\Sigma, \epsilon\rangle$ als das *Semiordnungsmonoid* über dem Spuralphabet $\Sigma$, wobei dieser Begriff durch das folgende Theorem fundiert wird. □

I.3.45. THEOREM. *$\mathscr{S}(\Sigma)$ ist ein Monoid.*

BEWEIS. Theorem I.3.41.(a) und (b). □

I.3.46. DEFINITION. Die Abbildungen $\mathrm{tr}_\Sigma : \mathbf{LSO}(\Sigma) \to \mathscr{P}(A_\Sigma^*)$ und $\mathrm{ord}_\Sigma : \mathbf{TR}(\Sigma) \to \mathbf{LSO}(\Sigma)$ sind durch

$$\mathrm{tr}_\Sigma =_{\mathrm{Def}} \mathrm{lin} \restriction \mathbf{LSO}(\Sigma) \text{ und } \mathrm{ord}_\Sigma([\sigma]_\Sigma) =_{\mathrm{Def}} \gamma(\langle\sigma\rangle_\Sigma)$$

definiert. □

Für $\sigma, \rho \in A_\Sigma^*$ gilt lediglich die Implikation $\sigma \approx_\Sigma \rho \Rightarrow \langle\sigma\rangle_\Sigma \equiv \langle\rho\rangle_\Sigma$ (s. den Beweis des folgenden Lemmas I.3.47 für eine Begründung), i.d.R. sind $\langle\sigma\rangle_\Sigma$ und $\langle\rho\rangle_\Sigma$ jedoch ungleich. Um für $\mathrm{ord}_\Sigma$ dennoch eine Definition zu erhalten, die unabhängig von der Wahl des Repräsentanten $\rho \in [\sigma]_\Sigma$ einer Spur $[\sigma]_\Sigma$ ist, wurde der kanonische Repräsentant der Äquivalenzklasse $\langle\sigma\rangle_\Sigma$ als Ergebnis von $\mathrm{ord}_\Sigma([\sigma]_\Sigma)$ gewählt.

I.3.47. LEMMA. *Sei $\Sigma$ ein Spuralphabet.*

(a) *Für alle $x \in \mathbf{LSO}(\Sigma)$ ist $\mathrm{tr}_\Sigma(x) \in \mathbf{TR}(\Sigma)$.*
(b) *$\mathrm{ord}_\Sigma$ ist wohldefiniert.*

BEWEIS. (a) Sei $\sigma \in \mathrm{lin}(x)$. $\mathrm{tr}_\Sigma(x) \subseteq [\sigma]_\Sigma$ folgt mit Lemma I.3.25, $[\sigma]_\Sigma \subseteq \mathrm{tr}_\Sigma(x)$ begründet sich so: Wenn $\sigma \approx_\Sigma \rho$ gilt, so gibt es eine Folge $\sigma_0, \sigma_1, \ldots, \sigma_n$ mit $\sigma_0 = \sigma$, $\sigma_n = \rho$ und $\sigma_i \sim_\Sigma \sigma_{i+1}$ für $0 \leqslant i < n$. Dabei ist $\sigma_i = \tau_1 ab \tau_2$ und $\sigma_{i+1} = \tau_1 ba \tau_2$ für $\tau_1, \tau_2 \in A_\Sigma^*$ und $a, b \in A_\Sigma$ mit $a\ I_\Sigma\ b$. Dann gilt jedoch $\tau_1(a \times b)\tau_2 \preccurlyeq \sigma_i$ sowie $\tau_1(a \times b)\tau_2 \preccurlyeq \sigma_{i+1}$. Mit Theorem I.3.37, (a) und (b) folgt $\langle\sigma_i\rangle_\Sigma \equiv x \equiv \langle\sigma_{i+1}\rangle_\Sigma$ für $0 \leqslant i < n$. Wir schließen $\sigma_i \in \mathrm{lin}(x)$ für $0 \leqslant i < n$ und insbesondere $\rho \in \mathrm{tr}_\Sigma(x)$.

(b) Sei $[\sigma]_\Sigma \in \mathbf{TR}(\Sigma)$ und seien $\rho \in [\sigma]_\Sigma$. Falls $\rho \sim_\Sigma \sigma$ gilt, gibt es $\sigma_1, \sigma_2 \in A_\Sigma^*$ und $a, b \in A_\Sigma$ mit $a\ I_\Sigma\ b$, so daß $\sigma = \sigma_1 ab \sigma_2$ und $\rho = \sigma_1 ba \sigma_2$ sind. Dann existieren jedoch Semiordnungen $x \in \mathbf{CSO}(\Sigma)$ mit $x \preccurlyeq \sigma$ und $x \preccurlyeq \rho$; etwa können wir $x \equiv \sigma_1(a \times b)\sigma_2$ wählen. Mit Theorem I.3.37.(a) folgt nun $\langle\sigma\rangle_\Sigma \equiv \langle\rho\rangle_\Sigma$, und mit Theorem I.2.29.(e) weiterhin $\gamma(\langle\sigma\rangle_\Sigma) = \gamma(\langle\rho\rangle_\Sigma)$.

Wenn $\sigma \approx_\Sigma \rho$, jedoch $\neg(\sigma \sim_\Sigma \rho)$ gilt, so gibt es eine Folge $\sigma_0, \sigma_1, \ldots, \sigma_{n-1}$ mit $\sigma_0 = \sigma$, $\sigma_{n-1} = \rho$ und $\sigma_i \sim_\Sigma \sigma_{i+1}$ für $0 \leqslant i < n-1$. Es folgt $\langle\sigma_0\rangle_\Sigma \equiv \langle\sigma_1\rangle_\Sigma \equiv \cdots \equiv \langle\sigma_{n-1}\rangle_\Sigma$, und wie eben $\gamma(\langle\sigma\rangle_\Sigma) = \gamma(\langle\rho\rangle_\Sigma)$ mit Theorem I.2.29.(e) □

I.3.48. THEOREM. *Ist $\Sigma$ ein Spuralphabet, so gilt $\mathrm{tr}_\Sigma \circ \mathrm{ord}_\Sigma = \mathrm{id}_{\mathbf{TR}(\Sigma)}$ und $\mathrm{ord}_\Sigma \circ \mathrm{tr}_\Sigma = \gamma \restriction \mathbf{LSO}(\Sigma)$.*

BEWEIS. Wir rechnen

$$\begin{aligned}\mathrm{tr}_\Sigma(\mathrm{ord}_\Sigma([\sigma]_\Sigma)) &= \mathrm{tr}_\Sigma(\gamma(\langle\sigma\rangle_\Sigma)) \\ &= \mathrm{lin}(\gamma(\langle\sigma\rangle_\Sigma)) \\ &= [\sigma]_\Sigma. \qquad \text{(Lemma I.3.47.(a), } \sigma \in \mathrm{lin}(\langle\sigma\rangle_\Sigma))\end{aligned}$$

Weiterhin gilt

$$\begin{aligned}\mathrm{ord}_\Sigma(\mathrm{tr}_\Sigma(x)) &= \mathrm{ord}_\Sigma(\mathrm{lin}(x)) \\ &= \gamma(\langle\sigma\rangle_\Sigma) \text{ für ein } \sigma \in \mathrm{lin}(x) \\ &= \gamma(x). \qquad (x \in \mathbf{LSO}(\Sigma))\end{aligned}$$

□

Da isomorphe Semiordnungen $x$ und $y$ auf identische Spuren $\mathrm{tr}_\Sigma(x) = \mathrm{tr}_\Sigma(y)$ abgebildet werden (dies folgt sofort aus Lemma I.2.9.(b) und der Definition I.3.46 von $\mathrm{tr}_\Sigma$), sind die Monoide $\mathscr{M}(\Sigma)$ und $\mathscr{S}(\Sigma)$ nicht isomorph. Wir können jedoch durch die folgende Konstruktion isomorphe Strukturen gewinnen:

(a) Die Struktur $\mathscr{S}^*(\Sigma) = \langle \mathbf{LSW}(\Sigma), \circ_\Sigma, \boldsymbol{\epsilon}\rangle$ ist ebenfalls ein Monoid, da $\equiv$ eine Kongruenz auf $\mathscr{S}(\Sigma)$ ist.

(b) Die Struktur $\mathscr{S}^*_\gamma(\Sigma) = \langle\{\gamma(x) \in \Gamma(A_\Sigma) : \boldsymbol{x} \in \mathbf{LSW}(\Sigma)\}, \circ_\Sigma, \epsilon\rangle$ ist ein Repräsentantensystem von $\mathscr{S}^*(\Sigma)$ und damit isomorph zu $\mathscr{S}^*(\Sigma)$.

I.3.49. KORROLAR. *Die Monoide $\mathscr{M}(\Sigma)$, $\mathscr{S}^*(\Sigma)$ und $\mathscr{S}^*_\gamma(\Sigma)$ sind isomorph.*

BEWEIS. Sowohl $\mathrm{tr}^*_\Sigma : \mathbf{LSW}(\Sigma) \to \mathbf{TR}(\Sigma)$ mit $\mathrm{tr}^*_\Sigma(\boldsymbol{x}) =_{\mathrm{Def}} \mathrm{tr}_\Sigma(x)$ als auch $\mathrm{ord}^*_\Sigma : \mathbf{TR}(\Sigma) \to \mathbf{LSW}(\Sigma)$ mit $\mathrm{ord}^*_\Sigma(\sigma) = [\mathrm{ord}_\Sigma(\sigma)]$ sind Isomorphismen. □

**Wohlgeformte Spursystem.** Wir beenden diesen Abschnitt mit einigen weiteren Definitionen, die wir im folgenden benötigen werden. Unter einem reduzierten Transitionssystem verstehen wir ein Transitionssystem, das keine vom Initialzustand nicht erreichbaren Zustände enthält. Ein Spursystem heißt *initialisiert*, wenn es eine ausgezeichnete Initialisierungsaktion enthält.

I.3.50. DEFINITION (Reduziertes Transitionssystem). Sei $\mathscr{T}$ ein Transitionssystem über einem Alphabet $A$. Wie nennen $\mathscr{T}$ *reduziert*, wenn für alle $s \in S_\mathscr{T}$ ein $\sigma \in A^*$ mit $s_\mathscr{T} \xrightarrow[\mathscr{T}]{\sigma} s$ existiert. □

I.3.51. DEFINITION (Initialisiertes Spursystem). Sei $\mathscr{T}$ ein Spursystem über $\Sigma$. Wir nennen $\mathscr{T}$ durch ein als *Initialisierungsaktion* bezeichnetes Symbol $a_I \in A_\Sigma$ *initialisiert*, wenn die folgenden Eigenschaften erfüllt sind:

(a) $s_\mathscr{T} \xrightarrow[\mathscr{T}]{a} s \Rightarrow a = a_I$ für alle $a \in A_\Sigma, s \in S_\mathscr{T}$,

(b) $s \xrightarrow[\mathscr{T}]{a_I} s' \Rightarrow s = s_\mathscr{T}$ für alle $s, s' \in S_\mathscr{T}$,

(c) $\neg s \xrightarrow[\mathscr{T}]{a} s_\mathscr{T}$ für alle $a \in A_\Sigma, s \in S_\mathscr{T}$.

$\mathcal{T}$ heißt *initialisiert*, wenn es eine Aktion $a_I \in A_\Sigma$ gibt, die $\mathcal{T}$ initialisiert. □

I.3.52. Beispiel (Netzsysteme). Ist $\mathcal{N}$ ein Netzsystem, so kann $\mathcal{N}$ zu einem Netzsystem $\mathcal{N}^*$ modifiziert werden, dessen induziertes Transitionssystem initialisiert ist: $\mathcal{N}^*$ entsteht in der folgenden Weise aus $\mathcal{N}$:

(a) Es wird eine Transition $t_I$ und ein Platz $p_I$ zu $\mathcal{N}$ hinzugefügt. $t_I$ (die Initialisierungsaktion) ist dabei in $T_\mathcal{N}$ nicht enthalten, ebenso gilt $p_I \notin P_\mathcal{N}$.
(b) Es werden die Kanten $p_I \longrightarrow t_I$ und $t_I \longrightarrow p$ für alle initial markierten Plätze $p \in Q_\mathcal{N}$ zu $\mathcal{N}$ hinzugefügt.
(c) Schließlich wird $p_I$ markiert, während alle weiteren Marken aus $\mathcal{N}^*$ entfernt werden, d. h. der Initialzustand von $\mathcal{N}^*$ ist $\{p_I\}$.

Anders ausgedrückt: $\mathcal{N}$ wird um eine Transition $t_I$ erweitert, die genau einmal, nämlich bei dem neu definierten Initialzustand $Q_{\mathcal{N}^*}$ konzessioniert ist. Schaltet $t_I$, so wird der ursprüngliche Initialzustand $Q_\mathcal{N}$ hergestellt.

Offenbar verändert diese Modifikation das Verhalten von $\mathcal{N}$ nur unwesentlich. Es gilt $\boldsymbol{t_I x} \in \mathbf{SL}(\mathcal{T}(\mathcal{N}^*)) \Leftrightarrow \boldsymbol{x} \in \mathbf{SL}(\mathcal{T}(\mathcal{N}))$. □

I.3.53. Definition (Initialisierung). Wir lernen aus Beispiel I.3.52, wie wir ein Spursystem $\mathcal{T}$ über $\Sigma$ initialisieren können. Sei $a_I \notin A_\Sigma$ und $s_0 \notin S_\mathcal{T}$. Wir konstruieren ein Spursystem $\mathcal{T}^*$ über einen Spuralphabet $\Sigma^*$ wie folgt:

(a) $A_{\Sigma^*} =_{\text{Def}} A_\Sigma \cup \{a_I\}$,
(b) $D_{\Sigma^*} =_{\text{Def}} D_\Sigma \cup D$, mit

$$a_I \; D \; a \;\&\; a \; D \; a_I \Leftrightarrow_{\text{Def}} a = a_I \vee \exists s \in S_\mathcal{T} \left( s \xrightarrow[\mathcal{T}]{a} s_\mathcal{T} \right)$$

und $I_{\Sigma^*} =_{\text{Def}} \bar{D}_{\Sigma^*}$,
(c) $S_{\mathcal{T}^*} =_{\text{Def}} S_\mathcal{T} \cup \{s_0\}$,
(d) $s_{\mathcal{T}^*} = s_0$,
(e) $\delta_{\mathcal{T}^*} = \delta_\mathcal{T} \cup \{\langle \langle s_0, a_I \rangle, s_\mathcal{T} \rangle\}$.

$\mathcal{T}^*$ wird *initialisiertes Spursystem* zu $\mathcal{T}$ genannt. □

I.3.54. Lemma. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$. Dann ist $\mathcal{T}$ mit $a_I$ initialisiert gdw. jedes $\boldsymbol{x} \in \mathbf{SL}(\mathcal{T})$ mit $\boldsymbol{x} \neq \boldsymbol{\epsilon}$ von der Form $\boldsymbol{x} = \boldsymbol{a_I y}$ ist, wobei $a_I \notin \lambda(y)$ gilt.*

Beweis. ($\Rightarrow$) Sei $\mathcal{T}$ durch $a_I$ initialisiert und sei $\boldsymbol{x} \in \mathbf{SL}(\mathcal{T})$, $\boldsymbol{x} \neq \boldsymbol{\epsilon}$. Offenbar gibt es ein $e \in \min(x)$ mit $\lambda_x(e) = a_I$, da nach Def. I.3.51.(b) $s_\mathcal{T} \xrightarrow[\mathcal{T}]{a_I}$ gilt. Wegen Def. I.3.51.(a) folgt $\lambda_x(e') = a_I$ für alle $e' \in \min(x)$. Damit gilt $e = e'$ für alle $e' \in E_x$, da $x$ eine Semiordnung ist. Def. I.3.51.(c) schließlich impliziert $e <_x e' \Rightarrow \lambda_x(e') \neq a_I$: Wir erhalten $\boldsymbol{y} = [x[E_x - \{e\}]]$.

($\Leftarrow$) Offensichtlich. □

I.3.55. Definition. Ein endliches, reduziertes und initialisiertes Spursystem über $\Sigma$ heißt *wohlgeformt*. □

KAPITEL II

# Komponenten und verteilte Systeme

In diesem Kapitel stellen wir Überlegungen zur Struktur nebenläufiger Systeme an. Wir gehen zunächst in Abschnitt II.1 davon aus, daß jedes Spursystem syntaktisch als Komposition *sequentieller Komponenten* beschrieben werden kann. Die lokalen Zustände solcher Komponenten bezeichnen wir als *Positionen*; Zuständen eines Spursystems werden Mengen solcher Positionen zugewiesen.

Unsere Definition einer Positionszuweisung erscheint zunächst *ad-hoc*, ist sie doch der Zustandsdefinition und der Schaltregel für Netzsysteme sehr ähnlich und damit anscheinend nicht auf beliebige nebenläufige System anwendbar. Wir zeigen jedoch in Abschnitt II.2, daß jedes Spursystem als Kompositum sequentieller Komponenten in Sinne von Abschnitt II.1 dargestellt werden kann.

Abschließend beschäftigen wir uns in Abschnitt II.3 mit dem Begriff des *verteilten Systems*. Wir geben drei Beispiele für Ausdrucksformen zur Beschreibung nebenläufiger Systeme an und zeigen Möglichkeiten auf, wie solche Systeme als verteilte Systeme interpretiert werden können.

## II.1. Positionszuweisungen und Komponenten

Viele nebenläufige Systeme ergeben sich als Komposition sequentieller Komponenten: Vorläufig sagen wir, eine solche sequentielle Komponente besteht aus einer Menge lokaler Zustände (im folgenden als *Positionen* bezeichnet) und von Transitionen, die Positionen einer oder mehrerer Komponenten in andere Positionen dieser Komponenten überführen. Wir setzen dabei nicht die Existenz einer CCS- oder CSP-artigen Kompositionsoperation voraus (obwohl dies sicher in vielen Fällen eine vernünftige Annahme ist).

Wir beginnen mit dem Begriff der *Positionszuweisung*. Den Zuständen eines Spursystems wird dabei eine Menge von *Positionen* zugeordnet, während mit den Aktionen des Systems ein Paar von Positionsmengen assoziiert wird, das den Positionswechsel der beeinflußten Komponenten beschreibt.

II.1.1. Definition. Sei $\Theta$ eine endliche Menge von *Positionen* und sei $\mathcal{T}$ ein Spursystem über $\Sigma$. Später definieren wir sogenannte Positionszuweisungen als Abbildungen der Form $\theta : (S_{\mathcal{T}} \to \mathcal{P}(\Theta)) \cup (A_\Sigma \to \mathcal{P}(\Theta) \times \mathcal{P}(\Theta))$, für solche

Abbildungen und für alle $a \in A_\Sigma$ mit $\theta(a) = \langle Q_1, Q_2 \rangle$ schreiben wir:

$$\dot{}\theta(a) =_{\text{Def}} Q_1,\ \theta^{\dot{}}(a) =_{\text{Def}} Q_2 \text{ und } \dot{\theta}(a) =_{\text{Def}} Q_1 \cup Q_2.$$

Mit diesen Schreibweisen nennen wir $\theta$ eine *Positionszuweisung* an $\mathcal{T}$, wenn für alle $a, b \in A_\Sigma$ und alle $s_1, s_2 \in S_\mathcal{T}$ die folgenden Eigenschaften erfüllt sind:

(a) $\dot{\theta}(a) \neq \varnothing$,
(b) $a\ I_\Sigma\ b \Rightarrow \dot{\theta}(a) \cap \dot{\theta}(b) = \varnothing$,
(c) $s_1 \xrightarrow[\mathcal{T}]{a} s_2 \Rightarrow \dot{}\theta(a) \subseteq \theta(s_1)\ \&\ \theta(s_2) = (\theta(s_1) - \dot{}\theta(a)) \cup \theta^{\dot{}}(a)$.

Eine Positionszuweisung $\theta$ heißt *kontaktfrei*, wenn

(d) $s \xrightarrow[\mathcal{T}]{a} \Rightarrow (\theta^{\dot{}}(a) - \dot{}\theta(a)) \cap \theta(s) = \varnothing$

gilt. Um den Schreibaufwand zu reduzieren, bezeichnen wir eine Positionszuweisung $\theta : (S_\mathcal{T} \to \mathcal{P}(\Theta)) \cup (A_\Sigma \to \mathcal{P}(\Theta) \times \mathcal{P}(\Theta))$ auch mit $\theta : \mathcal{T} \to \Theta$. □

II.1.1.(a) schließt triviale Positionszuweisungen der Form $\theta(a) = \langle \varnothing, \varnothing \rangle$ aus. Eigenschaft II.1.1.(b) stellt sicher, daß eine Positionszuweisung an die Zustände eines Spursystems mit der Unabhängigkeitsrelation dieses Spursystems verträglich ist. Eigenschaft II.1.1.(c) besagt, daß das Schalten von Systemaktionen konsistent zu der gewählten Positionszuweisung stattfindet.

Kontaktfreiheit (Eigenschaft II.1.1.(d)) ist eine weitere Konsistenzbedingung. Wir werden feststellen (Def. II.1.5), daß jede Positionszuweisung $\theta$ durch eine einfache Modifikation kontaktfrei gemacht werden kann.

Der Begriff der Positionszuweisungen verrät die Affinität des Autors dieser Arbeit zu den Petri-Netzen, wie wir durch das folgende Beispiel noch einmal explizit feststellen wollen:

II.1.2. Beispiel (Netzsysteme). Ist $\mathcal{N}$ ein Netzsystem und $\mathcal{T}(\mathcal{N})$ das zugehörige induzierte Spursystem, so gibt es offenbar eine Positionszuweisung $\theta_\mathcal{N} : \mathcal{T}(\mathcal{N}) \to P_\mathcal{N}$: Wir setzen $\theta_\mathcal{N}(Q) =_{\text{Def}} Q$ für alle $Q \in S_{\mathcal{T}(\mathcal{N})}$, sowie $\dot{}\theta_\mathcal{N}(t) =_{\text{Def}} {}^{\dot{}}t$ und $\theta^{\dot{}}_\mathcal{N}(t) =_{\text{Def}} t^{\dot{}}$ für alle $t \in T_\mathcal{N}$. □

II.1.3. Definition. Sei $\mathcal{T}$ ein Spursystem und $\boldsymbol{\Theta} = \{\Theta_i\}_{i \in J}$ eine Familie von endlichen und nicht leeren Positionsmengen für eine endliche und nicht leere Indexmenge $J$. Sei $\Theta = \bigcup_{i \in J} \Theta_i$. Wir sagen, $\mathcal{T}$ ist *$\boldsymbol{\Theta}$-überdeckt*, wenn es eine kontaktfreie Positionszuweisung $\theta : \mathcal{T} \to \Theta$ mit $|\theta(s) \cap \Theta_i| \leqslant 1$ für alle $s \in S_\mathcal{T}$ und für alle $i \in J$ gibt; jede der Mengen $\Theta_i$ nennen wir dann eine *Komponente*. Gibt es eine Familie von Positionen $\boldsymbol{\Theta} = \{\Theta_i\}_{i \in J}$, so daß $\mathcal{T}$ $\boldsymbol{\Theta}$-überdeckt ist, so nennen wir $\mathcal{T}$ *komponentenüberdeckt*. □

II.1.4. Beispiel. Sei $\mathcal{N}$ ein kontaktfreies Netzsystem und sei $\theta_\mathcal{N} : \mathcal{T}(\mathcal{N}) \to P_\mathcal{N}$ die zugehörige Positionszuweisung wie in Beispiel II.1.2 beschrieben. Wählen wir $\boldsymbol{\Theta}_\mathcal{N} =_{\text{Def}} \{\{p\}\}_{p \in P_\mathcal{N}}$, so ist $\mathcal{T}(\mathcal{N})$ $\boldsymbol{\Theta}$-überdeckt. □

Die Positionszuweisung $\theta_{\mathcal{N}}$ ist für kontaktbehaftete Netzsysteme nicht kontaktfrei. Wir können jedoch für jede Positionszuweisung eine kontaktfreie Positionszuweisung durch das Hinzufügen von *Komplementärpositionen* gewinnen.

II.1.5. DEFINITION. Sei $\theta : \mathcal{T} \to \Theta$ eine Positionszuweisung an ein Spursystem $\mathcal{T}$. Wir definieren eine Positionszuweisung $\theta^{\mathrm{c}}$ wie folgt:

Sei $(\cdot)^{\mathrm{c}} : p \mapsto p^{\mathrm{c}}$ eine Bijektion, die auf Positionen $p \in \Theta$ operiert, und sei

$$\Theta^{\mathrm{c}} =_{\mathrm{Def}} \{p^{\mathrm{c}} : p \in \Theta\}$$

eine Menge von *Komplementärpositionen* zu $\Theta$, wobei $\Theta \cap \Theta^{\mathrm{c}} = \varnothing$ gelte. Eine kontaktfreie Postionszuweisung $\theta^{\mathrm{c}} : \mathcal{T} \to \Theta \cup \Theta^{\mathrm{c}}$ kann nun wie folgt bestimmt werden:

(a) $\theta^{\mathrm{c}}(s) =_{\mathrm{Def}} \theta(s) \cup \{p^{\mathrm{c}} \in \Theta^{\mathrm{c}} : p \notin \theta(s)\}$ für alle $s \in S_{\mathcal{T}}$,
(b) ${}^{\cdot}\theta^{\mathrm{c}}(a) =_{\mathrm{Def}} {}^{\cdot}\theta(a) \cup \{p^{\mathrm{c}} \in \Theta^{\mathrm{c}} : p \in \theta^{\cdot}(a) - {}^{\cdot}\theta(a)\}$ und
(c) $\theta^{\mathrm{c}\cdot}(a) =_{\mathrm{Def}} \theta^{\cdot}(a) \cup \{p^{\mathrm{c}} \in \Theta^{\mathrm{c}} : p \in {}^{\cdot}\theta(a) - \theta^{\cdot}(a)\}$ für alle $a \in A_{\Sigma}$.

□

Wir verzichten darauf nachzuweisen, daß $\theta^{\mathrm{c}}$ tatsächlich eine kontaktfreie Positionszuweisung ist: Der Beweis besteht in der direkten Anwendung der entsprechenden Definitionen. Ebenso ist leicht zu sehen, daß jedes Spursystem $\mathcal{T}$ $\boldsymbol{\Theta}^{\mathrm{c}}$-überdeckt ist, wenn wir $\boldsymbol{\Theta}^{\mathrm{c}} =_{\mathrm{Def}} \{\{p, p^{\mathrm{c}}\}\}_{p \in \Theta}$ wählen.

II.1.6. LEMMA. *Ist $\theta : \mathcal{T} \to \Theta$ eine Positionszuweisung an ein Spursystem $\mathcal{T}$, so gibt es eine Familie von Positionsmengen $\boldsymbol{\Theta} = \{\Theta_i\}_{i \in J}$ für eine endliche Indexmenge $J$, so daß $\mathcal{T}$ $\boldsymbol{\Theta}$-überdeckt ist.*

Definition II.1.1.(a) fordert $\dot{\theta}(a) \neq \varnothing$ für alle Aktionen $a$. Definieren wir für ein Spursystem $\mathcal{T}$ über $\Sigma$ die folgende triviale Positionszuweisung $\iota : \mathcal{T} \to \Theta_0$, indem wir zunächst für alle $a \in A_{\Sigma}$ eine Position $p_a$ einführen; die Abbildung $a \mapsto p_a$ sei dabei injektiv. Wir setzen $\Theta_0 =_{\mathrm{Def}} \{p_a : a \in A_{\Sigma}\}$, $\iota(a) =_{\mathrm{Def}} \langle\{p_a\}, \{p_a\}\rangle$ für alle $a \in A_{\Sigma}$ und $\iota(s) = \Theta_0$ für alle $s \in S_{\mathcal{T}}$. Für diese Positionszuweisung existiert genau eine Familie von Positionsmengen $\boldsymbol{\Theta}_0$, so daß $\mathcal{T}$ $\boldsymbol{\Theta}_0$-überdeckt ist, nämlich $\boldsymbol{\Theta}_0 = \{\{p_a\}\}_{a \in A_{\Sigma}}$.

Eine solche triviale Positionszuweisung ist für die Zwecke der beiden folgenden Kapitel nicht nützlich: Wir verwenden Positionszuweisungen in Kapitel III zur Verbesserung der Laufzeit des Erzeugungsalgorithmus für Prozeßautomaten und in Kapitel IV zur Definition der temporalen Logik DCTL. Unter Verwendung der Positionszuweisung $\iota$ haben die in Abschnitt III.3 diskutierten Verfahren keinerlei Effekt, während in DCTL lediglich triviale Aussagen zur Schaltfähigkeit von Systemaktionen möglich sind. Allerdings wollen wir Positionszuweisungen der Form $\theta(a) = \langle\{p\}, \{p\}\rangle$ an *einige* Aktionen $a \in A_{\Sigma}$ nicht von vornherein ausschließen, da ein nebenläufiges System durchaus Komponenten der Form $\{p\}$ enthalten darf.

Wir führen nun zwei Anforderungen an Positionszuweisungen $\theta$ ein, die sicherstellen, daß $\theta$ die Kausalstruktur eines nebenläufigen Systems reflektiert.

II.1.7. DEFINITION (Kausal- und Konfliktvollständigkeit). Sei $\mathcal{T}$ ein Spursystem über $\Sigma$ und sei $\Theta$ eine Menge von Positionen. Eine Positionszuweisung $\theta : \mathcal{T} \to \Theta$ heist

(a) *konfliktvollständig*, wenn für alle Aktionen $a, b \in A_\Sigma$ mit $a \neq b$ und $a\ D_\Sigma\ b$ und für alle $s \in S_\mathcal{T}$

$$a \in \mathrm{en}_\mathcal{T}(s)\ \&\ b \in \mathrm{en}_\mathcal{T}(s) \Rightarrow \dot{\theta}(a) \cap \dot{\theta}(b) \neq \varnothing$$

gilt, und

(b) *kausalvollständig*, wenn für alle Aktionen $a, b \in A_\Sigma$ mit $a \neq b$ und $a\ D_\Sigma\ b$ und für alle $s_1, s_2 \in S_\mathcal{T}$ die Implikation

$$s_1 \xrightarrow[\mathcal{T}]{a} s_2\ \&\ s_2 \xrightarrow[\mathcal{T}]{b} \Rightarrow \dot{\theta}(a) \cap \dot{\theta}(b) \neq \varnothing$$

erfüllt ist.

□

Wir fordern aber nicht, daß jede im folgenden verwendete Positionszuweisung $\theta$ konflikt- oder kausalvollständig ist. Zwar ist die Positionszuweisung $\theta_\mathcal{N} : \mathcal{T}(\mathcal{N}) \to P_\mathcal{N}$ an das durch ein Netzsystem induzierte Spursystem offenbar konflikt- und kausalvollständig, allerdings gibt es Beschreibungsformen für nebenläufige Systeme, aus denen nicht ohne weiteres eine konflikt- oder kausalvollständige Positionszuweisung allein durch strukturelle Analysen (d. h. ohne explizite Erzeugung eines vollständigen Spursystems) abgeleitet werden kann.

## II.2. Vollständigkeit des Komponentenbegriffs

In diesem Abschnitt befassen wir uns mit der Fragestellung, ob für jedes Spursystem eine Positionszuweisung bzw. eine Komponentenüberdeckung gefunden werden kann. Vom Standpunkt des Petri-Netz-Theoretikers ist weiterhin interessant, unter welchen Voraussetzungen das durch eine Positionszuweisung $\theta : \mathcal{T} \to \Theta$ induzierte Petri-Netz $\mathcal{N}(\mathcal{T}, \theta)$ die Struktur von $\mathcal{T}$ vollständig beschreibt, d. h. unter welchen Bedingungen $\mathcal{T}(\mathcal{N}(\mathcal{T}, \theta))$ isomorph zu $\mathcal{T}$ ist.

Die Begriffe und Ergebnisse, die in diesem Abschnitt präsentiert werden, sind eng verbunden mit der Theorie der *Regionen* von Transitions- bzw. Spursystemen, insbesondere mit den Arbeiten von Nielsen und Winskel, wir halten uns in unserer Darstellung eng an [**62**].

Eine *Region* in einem Spursystem $\mathcal{T}$ über $\Sigma$ ist eine Menge von Tupeln der Form $\langle s_1, a, s_2 \rangle$,[1] so daß $s_1 \xrightarrow[\mathcal{T}]{a} s_2$ gilt. Daneben lassen wir noch Tupel $\langle s, \bullet, s \rangle$, als Elemente von Regionen zu, wobei $s \in S_\mathcal{T}$ und $\bullet \notin A_\Sigma$ ist. Regionen werden als Positionen Verwendung finden.

[1] Im Zusammenhang mit elementaren Netzsystemen werden Regionen üblicherweise als Zustandsmengen definiert. Die Verwendung von Tupeln, die Zustandsübergänge bezeichnen, ist für unsere Zwecke jedoch geigneter.

Die zur Definition der Positionszuweisung analoge Schaltregel für Netzsysteme soll uns hier helfen, die folgende Definition von Regionen zu motivieren. Sei $\mathcal{N}$ ein Netzsystem und sei $p \in P_{\mathcal{N}}$. Eine Region für $p$ umfaßt nun all diejenigen Zustandswechsel, bei denen $p$ ununterbrochen markiert ist; wir bezeichnen diese Menge mit $\mathrm{ext}(p)$:

$$\begin{aligned}&\langle Q_1, t, Q_2\rangle \in \mathrm{ext}(p)\\ &\quad \Leftrightarrow_{\mathrm{Def}} \quad \left(Q_1 \xrightarrow[\mathcal{N}]{t} Q_2 \;\&\; p \notin \dot{t} \vee Q_1 = Q_2 \;\&\; t = \bullet\right) \;\&\; p \in Q_1 \cap Q_2.\end{aligned}$$

Ist $p \in {}^{\cdot}t_1$, so ist $\langle Q_1, \bullet, Q_1\rangle \in \mathrm{ext}(p)$ für alle Zustände $Q_1, Q_2$ mit $Q_1 \xrightarrow[\mathcal{N}]{t_1} Q_2$, insbesondere ist jedoch $\langle Q_1, t_1, Q_2\rangle \notin \mathrm{ext}(p)$, da $p$ nach dem Feuern von $t_1$ nicht mehr markiert ist (oder $p$ ist ebenso in $t_1^{\cdot}$ enthalten; diesen Fall wollen wir als momentane Unterbrechung der Markiertheit von $p$ werten). Dann gilt jedoch auch $\langle Q_1', t_1, Q_2'\rangle \notin \mathrm{ext}(p)$ für jeden weiteren Zustandsübergang $Q_1' \xrightarrow[\mathcal{N}]{t_1} Q_2'$:

Analoge Überlegungen können für den Fall $p \in t_1^{\cdot}$ angestellt werden. In $\mathcal{T}(\mathcal{N})$ ist ein Platz $p \in {}^{\cdot}t_1$ also dadurch charakterisiert, das für jedes Tupel der Form $\langle Q_1, t_1, Q_2\rangle$

$$\langle Q_1, \bullet, Q_1\rangle \in \mathrm{ext}(p) \text{ und } \langle Q_1, t_1, Q_2\rangle \notin \mathrm{ext}(p)$$

gilt. Ebenso erhalten wir im Fall $p \in t_1^{\cdot}$ für jedes Tupel der Form $\langle Q_1, t_1, Q_2\rangle$

$$\langle Q_1, t_1, Q_2\rangle \notin \mathrm{ext}(p) \text{ und } \langle Q_2, \bullet, Q_2\rangle \in \mathrm{ext}(p).$$

Tupel der Form $\langle Q, \bullet, Q\rangle$ repräsentieren also Marken.

II.2.1. Definition. Sei $\mathcal{T}$ ein Spursystem über $\Sigma$. Sei $\bullet \notin A_\Sigma$. Wir setzen

$$\begin{aligned}\Delta_{\mathcal{T}} &=_{\mathrm{Def}} \left\{\langle s_1, a, s_2\rangle \in S_{\mathcal{T}} \times A_\Sigma \times S_{\mathcal{T}} : s_1 \xrightarrow[\mathcal{T}]{a} s_2\right\} \text{ und}\\ \Delta_{\mathcal{T}}^{\bullet} &=_{\mathrm{Def}} \Delta_{\mathcal{T}} \cup \left\{\langle s_1, \bullet, s_2\rangle \in S_{\mathcal{T}} \times \{\bullet\} \times S_{\mathcal{T}} : s_1 = s_2\right\}.\end{aligned}$$

Für jede Teilmenge $R \subseteq \Delta_{\mathcal{T}}^{\bullet}$ definieren wir

$$\begin{aligned}{}^{\cdot}R &=_{\mathrm{Def}} \left\{\langle s_1, a, s_2\rangle \in \Delta_{\mathcal{T}} : \langle s_1, a, s_2\rangle \notin R \;\&\; \langle s_2, \bullet, s_2\rangle \in R\right\},\\ R^{\cdot} &=_{\mathrm{Def}} \left\{\langle s_1, a, s_2\rangle \in \Delta_{\mathcal{T}} : \langle s_1, \bullet, s_1\rangle \in R \;\&\; \langle s_1, a, s_2\rangle \notin R\right\} \text{ und}\\ \dot{R} &=_{\mathrm{Def}} {}^{\cdot}R \cup R^{\cdot}\end{aligned}$$

Eine nicht leere Teilmenge $R \subseteq \Delta_{\mathcal{T}}^{\bullet}$ wird als *Region* in $\mathcal{T}$ bezeichnet, wenn $R$ die folgenden Bedingungen erfüllt:

(a) $\langle s_1, a, s_2\rangle \in R \Rightarrow \langle s_1, \bullet, s_1\rangle \in R \;\&\; \langle s_2, \bullet, s_2\rangle \in R$,
(b) $\langle s_1, a, s_2\rangle \in {}^{\cdot}R \;\&\; \langle s_1', a, s_2'\rangle \in \Delta_{\mathcal{T}} \Rightarrow \langle s_1', a, s_2'\rangle \in {}^{\cdot}R$,
$\langle s_1, a, s_2\rangle \in R^{\cdot} \;\&\; \langle s_1', a, s_2'\rangle \in \Delta_{\mathcal{T}} \Rightarrow \langle s_1', a, s_2'\rangle \in R^{\cdot}$,
(c) $\langle s_1, a, s_2\rangle \in \dot{R} \;\&\; \langle s_3, b, s_4\rangle \in \dot{R} \Rightarrow a\ D_\Sigma\ b$.

Mit $\mathbf{REG}(\mathcal{T})$ bezeichnen wir die Menge der Regionen in $\mathcal{T}$, während

$$\mathrm{reg}_{\mathcal{T}}(s) =_{\mathrm{Def}} \{R \in \mathbf{REG}(\mathcal{T}) : \langle s, \bullet, s\rangle \in R\}$$

für die Regionen eines Zustandes $s \in S_{\mathcal{T}}$ steht. Schließlich setzen wir für alle $a \in A_\Sigma$

$$\begin{aligned}
\dot{\mathrm{reg}}_{\mathcal{T}}(a) &=_{\mathrm{Def}} \{R \in \mathbf{REG}(\mathcal{T}) : \exists s_1, s_2 \in S_{\mathcal{T}}(\langle s_1, a, s_2\rangle \in R^{\cdot})\}\,, \\
\mathrm{reg}^{\cdot}_{\mathcal{T}}(a) &=_{\mathrm{Def}} \{R \in \mathbf{REG}(\mathcal{T}) : \exists s_1, s_2 \in S_{\mathcal{T}}(\langle s_1, a, s_2\rangle \in \dot{R})\} \text{ und} \\
\dot{\mathrm{reg}}_{\mathcal{T}}(a) &=_{\mathrm{Def}} \dot{\mathrm{reg}}_{\mathcal{T}}(a) \cup \mathrm{reg}^{\cdot}_{\mathcal{T}}(a)
\end{aligned}$$

□

II.2.2. Theorem. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$. Unter der Voraussetzung, daß für alle $a \in A_\Sigma$ ein Zustand $s \in S_{\mathcal{T}}$ mit $a \in \mathrm{en}_{\mathcal{T}}(s)$ existiert, ist die Abbildung* $\mathrm{reg}_{\mathcal{T}} : (S_{\mathcal{T}} \to \mathbf{REG}(\mathcal{T})) \cup (A_\Sigma \to \mathbf{REG}(\mathcal{T}) \times \mathbf{REG}(\mathcal{T}))$ *mit*

$$\mathrm{reg}_{\mathcal{T}}(a) =_{Def} \langle \dot{\mathrm{reg}}_{\mathcal{T}}(a), \mathrm{reg}^{\cdot}_{\mathcal{T}}(a)\rangle \qquad (a \in A_\Sigma)$$

*eine kontaktfreie –, kausal- und konfliktvollständige Positionszuweisung* $\mathrm{reg}_{\mathcal{T}} : \mathcal{T} \to \mathbf{REG}(\mathcal{T})$.

Beweis. Wir haben die Eigenschaften (a), (b), (c) und (d) aus Definition II.1.1 nachzuweisen.

(a) Sei $a \in A_\Sigma$. Setzen wir

$$\Delta =_{\mathrm{Def}} \{\langle s, b, s'\rangle \in \Delta_{\mathcal{T}} : a \neq b\}$$

und

$$R =_{\mathrm{Def}} \Delta \cup \{\langle s, \bullet, s\rangle : \exists s' \in S_{\mathcal{T}}, b \in A_\Sigma\, (\langle s, b, s'\rangle \in \Delta \vee \langle s', b, s\rangle \in \Delta)\}\,.$$

Wir zeigen zunächst, daß $R$ eine Region in $\mathcal{T}$ ist: Zunächst stellen wir fest, daß aufgrund der Voraussetzung des Theorems $R \neq \varnothing$ gilt. Die in Def. II.2.1 geforderten Eigenschaften (a) bis (c) sind offenbar erfüllt. Seien $s_1, s_2 \in S_{\mathcal{T}}$ Zustände und sei $b \in A_\Sigma$ eine Aktion mit $a \neq b$, so daß $\langle s_1, b, s_2\rangle \in \Delta_{\mathcal{T}}$ gilt. Dann ist $\langle s_1, \bullet, s_1\rangle \in R$ und $\langle s_1, b, s_2\rangle \in R$, weiterhin rechnen wir:

$$\begin{aligned}
\langle s_1, \bullet, s_1\rangle \in R \,\&\, \langle s_1, b, s_2\rangle \in R \,\&\, a \neq b &\Rightarrow \langle s_1, \bullet, s_1\rangle \in R \,\&\, \langle s_1, a, s_2\rangle \notin R && \text{(Def. } R\text{)} \\
&\Rightarrow \langle s_1, a, s_2\rangle \in R^{\cdot} && \text{(Def } R^{\cdot}\text{)} \\
&\Rightarrow R \in \dot{\mathrm{reg}}(a). && \text{(Def } \dot{\mathrm{reg}}(a)\text{)}
\end{aligned}$$

$R \in \mathrm{reg}^{\cdot}(a)$ folgt analog.

(b) Nehmen wir $a\ I_\Sigma\ b$ an. Dann impliziert die Kontraposition von Definition II.2.1.(c), in Verbindung mit II.2.1.(b) $\dot{\mathrm{reg}}(a) \cap \dot{\mathrm{reg}}(b) = \varnothing$.

(c) Es gelte $s_1 \xrightarrow[\mathcal{J}]{a} s_2$, d. h. $\langle s_1, a, s_2\rangle \in \Delta_{\mathcal{J}}$. Um $\dot{\mathrm{reg}}_{\mathcal{J}}(a) \subseteq \mathrm{reg}_{\mathcal{J}}(s_1)$ nachzuweisen, rechnen wir:

$$\begin{aligned}
R \in \dot{\mathrm{reg}}_{\mathcal{J}}(a) &\Leftrightarrow \exists s_3, s_4 \in S_{\mathcal{J}}(\langle s_3, a, s_4\rangle \in R^{\cdot}) && \text{(Def. } \dot{\mathrm{reg}}_{\mathcal{J}}\text{)}\\
&\Rightarrow \langle s_1, a, s_2\rangle \in R^{\cdot} && \text{(Def. II.2.1.(b) für } R^{\cdot}\text{)}\\
&\Rightarrow \langle s_1, \bullet, s_1\rangle \in R \;\&\; \langle s_2, a, s_2\rangle \notin R && \text{(Def. } R^{\cdot}\text{)}\\
&\Rightarrow \langle s_1, \bullet, s_1\rangle \in R\\
&\Leftrightarrow R \in \mathrm{reg}_{\mathcal{J}}(s_1). && \text{(Def. } \mathrm{reg}_{\mathcal{J}}(s_1)\text{)}
\end{aligned}$$

$\dot{\mathrm{reg}}_{\mathcal{J}}(a) \subseteq \mathrm{reg}_{\mathcal{J}}(s_2)$ folgt ebenso. Weiterhin stellen wir

$$\mathrm{reg}_{\mathcal{J}}(s_1) - \dot{\mathrm{reg}}_{\mathcal{J}}(a) = \mathrm{reg}_{\mathcal{J}}(s_2) - \dot{\mathrm{reg}}_{\mathcal{J}}(a)$$

fest:

$$\begin{aligned}
R \in \mathrm{reg}_{\mathcal{J}}(s_1) - \dot{\mathrm{reg}}_{\mathcal{J}}(a) &\Leftrightarrow \langle s_1, \bullet, s_1\rangle \in R \;\&\; \langle s_1, a, s_2\rangle \in R\\
&&& \text{(Def. } \mathrm{reg}_{\mathcal{J}}(s_1), \dot{\mathrm{reg}}_{\mathcal{J}}(a)\text{)}\\
&\Leftrightarrow \langle s_2, \bullet, s_2\rangle \in R \;\&\; \langle s_1, a, s_2\rangle \in R && \text{(Def. II.2.1.(a))}\\
&\Leftrightarrow R \in \mathrm{reg}_{\mathcal{J}}(s_2) - \dot{\mathrm{reg}}_{\mathcal{J}}(a). && \text{(Def. } \mathrm{reg}_{\mathcal{J}}(s_2), \dot{\mathrm{reg}}_{\mathcal{J}}(a)\text{)}
\end{aligned}$$

Wir schießen $\mathrm{reg}_{\mathcal{J}}(s_2) = (\mathrm{reg}_{\mathcal{J}}(s_1) - \dot{\mathrm{reg}}_{\mathcal{J}}(a)) \cup \dot{\mathrm{reg}}_{\mathcal{J}}(a)$.

(d) Zur Nachweis der Kontaktfreiheit von $\mathrm{reg}_{\mathcal{J}}$ rechnen wir

$$\begin{aligned}
R \in \mathrm{reg}_{\mathcal{J}}(s_1) \cap \dot{\mathrm{reg}}_{\mathcal{J}}(a) &\Rightarrow \langle s_1, \bullet, s_1\rangle \in R\\
&\qquad \& \forall s_3, s_4 \in S_{\mathcal{J}} \left(s_3 \xrightarrow[\mathcal{J}]{a} s_4 \Rightarrow \langle s_3, a, s_4\rangle \in \dot{R}\right)\\
&\qquad \text{(Def. } \mathrm{reg}_{\mathcal{J}}(s)\text{, Def. } \dot{\mathrm{reg}}_{\mathcal{J}} \text{ sowie Def. II.2.1.(b) für } \dot{R}\text{)}\\
&\Rightarrow \langle s_1, \bullet, s_1\rangle \in R \;\&\; \langle s_1, a, s_2\rangle \in \dot{R} \qquad (\langle s_1, a, s_2\rangle \in \Delta_{\mathcal{J}})\\
&\Rightarrow \langle s_1, \bullet, s_1\rangle \in R \;\&\; \langle s_2, \bullet, s_2\rangle \in R \;\&\; \langle s_1, a, s_2\rangle \notin R\\
&&& \text{(Def. } \dot{R}\text{)}\\
&\Rightarrow \langle s_1, a, s_2\rangle \in R^{\cdot} && \text{(Def. } R^{\cdot}\text{)}\\
&\Rightarrow \forall s_3, s_4 \in S_{\mathcal{J}} \left(s_3 \xrightarrow[\mathcal{J}]{a} s_4 \Rightarrow \langle s_3, a, s_4\rangle \in R^{\cdot}\right)\\
&&& \text{(Def. II.2.1.(b) für } R^{\cdot}\text{)}\\
&\Rightarrow R \in \dot{\mathrm{reg}}_{\mathcal{J}}(a). && \text{(Def. } \dot{\mathrm{reg}}_{\mathcal{J}}(a)\text{)}
\end{aligned}$$

Um den Beweis von (d) abzuschließen, stellen wir die Gültigkeit der Implikation

$$\mathrm{reg}_{\mathcal{J}}(s_1) \cap \dot{\mathrm{reg}}_{\mathcal{J}}(a) \subseteq \dot{\mathrm{reg}}_{\mathcal{J}}(a) \Rightarrow \mathrm{reg}_{\mathcal{J}}(s_1) \cap (\dot{\mathrm{reg}}_{\mathcal{J}}(a) - \dot{\mathrm{reg}}_{\mathcal{J}}(a)) = \varnothing.$$

fest.

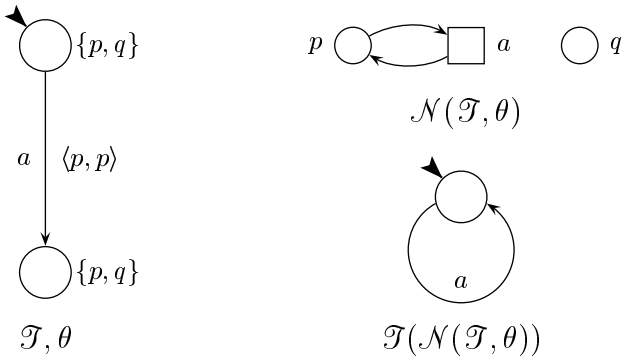



ABBILDUNG II.1. $\mathcal{T}$ und $\mathcal{T}(\mathcal{N}(\mathcal{T},\theta))$ sind i. Allg. nicht isomorph.

---

Zum Nachweis der Konfliktvollständigkeit. Seien $a, b \in A_\Sigma$ mit $a \neq b$ und $a\ D_\Sigma\ b$. Sei $s \in S_\mathcal{T}$ ein Zustand mit $a \in \mathrm{en}_\mathcal{T}(s)$ und $b \in \mathrm{en}_\mathcal{T}(s)$. Setzen wir

$$\Delta =_{\mathrm{Def}} \{\langle s, c, s'\rangle \in \Delta_\mathcal{T} : c \neq a \ \&\ c \neq b\}$$

und

$$R =_{\mathrm{Def}} \Delta \cup \{\langle s, \bullet, s\rangle : \exists s' \in S_\mathcal{T}, c \in A_\Sigma\ (\langle s, c, s'\rangle \in \Delta \vee \langle s', c, s\rangle \in \Delta)\}\,.$$

Dann ist $R$ eine Region mit $R \in {}^\cdot\mathrm{reg}(a) \cap {}^\cdot\mathrm{reg}(b)$, der Beweis ist analog zum Beweis von Eigenschaft (a). Dieselbe Konstruktion kann auch zum Nachweis der Kausalvollständigkeit verwendet werden. □

Offenbar gibt es für jedes Spursystem, für das eine Positionszuweisung existiert, ein Netzsystem:

II.2.3. DEFINITION. Sei $\mathcal{T}$ ein Spursystem über $\Sigma$ und $\theta : \mathcal{T} \to \Theta$ eine Positionszuweisung. Wir setzen

$$\mathcal{N}(\mathcal{T},\theta) =_{\mathrm{Def}} \langle \Theta, A_\Sigma, F, \theta(s_\mathcal{T})\rangle,$$

wobei die Flußrelation $F$ des Netzsystems $\mathcal{N}(\mathcal{T},\theta)$ implizit durch

$${}^\cdot a =_{\mathrm{Def}} {}^\cdot\theta(a) \text{ und } a^\cdot =_{\mathrm{Def}} \theta^\cdot(a)$$

definiert ist. $\mathcal{N}(\mathcal{T},\theta)$ heißt das durch $\mathcal{T}$ und $\theta$ *induzierte Netzsystem.* □

II.2.4. BEMERKUNG. $\mathcal{T}(\mathcal{N}(\mathcal{T},\theta))$ ist i. Allg. nicht isomorph zu $\mathcal{T}$, wie das Beispiel in Abbildung II.1 zeigt. $\theta$ verstößt gegen zwei *Separationseigenschaften*: Einerseits wird verschiedenen Zuständen in $\mathcal{T}$ dieselbe Positionsmenge zugeteilt, andererseits ist die Aktion $a$ bei dem einen Zustand von $\mathcal{T}$ konzessioniert, bei dem anderen jedoch nicht.

Wir können diese Verstöße auch so interpretieren: $\theta$ beschreibt zwar Positionen sequentieller Komponenten in $\mathcal{T}$, jedoch mag es weitere Komponenten (oder allgemeiner,

andere Mechanismen, die das Systemverhalten steuern) geben, deren explizite Einbeziehung eine Separation der beiden Zustände von $\mathcal{T}$ erlauben bzw. eine Erklärung für die Konzessioniertheit von $a$ bei nur einem der beiden Zustände liefern würde. □

Fragen wir uns nun, unter welchen Umständen eine Positionszuweisung die tatsächlichen Abhängigkeiten in einem Spursystem beschreibt. Das folgende Lemma besagt: Wenn wir eine Positionszuweisung $\theta$ an ein Spursystem $\mathcal{T}$ so wählen, daß sich das Verhalten von $\mathcal{T}$ vollständig aus $\theta$ ergibt, so handelt es sich bei $\mathcal{T}$ tatsächlich um das induzierte Spursystem eines Netzsystems.

II.2.5. LEMMA. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$ und sei $\theta : \mathcal{T} \to \Theta$ eine kontaktfreie Positionszuweisung. I. Wenn die folgenden Eigenschaften erfüllt sind, dann ist $\mathcal{T}(\mathcal{N}(\mathcal{T},\theta))$ isomorph zu $\mathcal{T}$:*

(a) *$\mathcal{T}$ ist reduziert.*
(b) *Für alle $a \in A_\Sigma$ gibt es einen Zustand $s \in S_\mathcal{T}$, so daß $a \in \mathrm{en}_\mathcal{T}(s)$ ist.*
(c) *$\theta(s) = \theta(s') \Rightarrow s = s'$ für alle $s, s' \in S_\mathcal{T}$.*
(d) *$\theta(a) \subseteq \theta(s) \Rightarrow s \xrightarrow[\mathcal{T}]{a}$ für alle $s \in S_\mathcal{T}$, $a \in A_\Sigma$.*

*II. Ist $\mathcal{N}$ ein Netzsystem, so gelten die obigen Eigenschaften für $\mathcal{T}(\mathcal{N})$ und $\theta_\mathcal{N}$.*

BEWEIS. I. Wir zeigen, daß die Abbildung $f =_{\mathrm{Def}} \theta \restriction S_\mathcal{T}$ eine Bijektion ist, die die Isomorphiebeziehung zwischen $\mathcal{T}$ und $\mathcal{T}(\mathcal{N}(\mathcal{T},\theta))$ herstellt.

(1) Nach Bedingung (c) ist $f$ injektiv.

(2) $s_1 \xrightarrow[\mathcal{T}]{a} s_2$ impliziert $f(s_1) \xrightarrow[\mathcal{N}(\mathcal{T},\theta)]{a} f(s_2)$ nach Definition II.1.1.(c), während zum Nachweis der Implikation

$$f(s_1) \xrightarrow[\mathcal{N}(\mathcal{T},\theta)]{a} f(s_2) \Rightarrow s_1 \xrightarrow[\mathcal{T}]{a} s_2$$

zusätzlich noch Bedingung (d) benötigt wird.

(3) $Q_{\mathcal{N}(\mathcal{T},\theta)} = f(s_\mathcal{T})$ gilt nach Definition von $\mathcal{N}(\mathcal{T},\theta)$.

(4) Zum Beweis der Surjektivität von $f$ ist es mit Eigenschaft (a) ausreichend nachzuweisen, daß

$$f(s_1) \xRightarrow[\mathcal{N}(\mathcal{T},\theta)]{\sigma} f(s_2) \Rightarrow s_1 \xRightarrow[\mathcal{T}]{\sigma} s_2$$

für alle $\sigma \in A_\Sigma$ gilt. Dies folgt jedoch mit Hilfe einer einfachen Induktion über der Länge von $\sigma$ unter Verwendung von (2) im Induktionsanfang und (3) im Induktionsschritt.

II. folgt direkt aus der Definition von $\mathcal{T}(\mathcal{N})$. □

II.2.6. KORROLAR. *Wenn $\mathrm{reg}_\mathcal{T}$ und $\mathcal{T}$ die Bedingungen* (a) *bis* (d) *aus Lemma II.2.5 erfüllen, ist $\mathcal{T}(\mathcal{N}(\mathcal{T},\mathrm{reg}_\mathcal{T}))$ isomorph zu $\mathcal{T}$.*

II.2.7. KORROLAR. *Sei $\theta : \mathcal{T} \to \Theta$ eine kontaktfreie Positionszuweisung an ein Spursystem $\mathcal{T}$ über $\Sigma$, so daß die unter Lemma II.2.5.I aufgeführten Eigenschaften erfüllt sind. Dann ist $\theta$ konflikt- und kausalvollständig.*

BEWEIS. Für jedes Netzsystem $\mathcal{N}$ ist die Positionszuweisungen $\theta_{\mathcal{N}} : \mathcal{T}(\mathcal{N}) \to P_{\mathcal{N}}$ konflikt- und kausalvollständig. Sei $f$ ein Isomorphismus von $\mathcal{T}(\mathcal{N}(\mathcal{T}, \theta))$ nach $\mathcal{T}$; $f$ existiert wegen Lemma II.2.5. Es gilt $\theta(s) = \theta_{\mathcal{N}(\mathcal{T},\theta)}(f(s))$ für alle $s \in S_{\mathcal{T}}$ und natürlich $\theta(a) = \theta_{\mathcal{N}(\mathcal{T},\theta)}(a)$ für alle $a \in A_\Sigma$. □

II.2.8. KORROLAR. *Für jedes reduzierte Spursystem $\mathcal{T}$ über einem Spuralphabet $\Sigma$ gibt es unter der Voraussetzung, daß für jedes $a \in A_\Sigma$ ein $s \in S_{\mathcal{T}}$ mit $a \in \mathrm{en}_{\mathcal{T}}(s)$ existiert, eine Familie $\boldsymbol{\Theta} = \{\Theta_i\}_{i\in J}$ von Positionen und eine Positionszuweisung $\theta : \mathcal{T} \to \bigcup_{i\in J} \Theta_i$, so daß $\theta$ kontaktfrei, konflikt- und kausalvollständig ist. Weiterhin ist $\mathcal{T}$ $\boldsymbol{\Theta}$-überdeckt.*

BEWEIS. Sofortige Folgerung aus der in Beispiel II.1.2 beschriebenen Konstruktion mit $\mathcal{N} = \mathcal{N}(\mathcal{T}, \mathrm{reg}_{\mathcal{T}})$ und Theorem II.2.2. □

## II.3. Verteilte Systeme — Definition und Beispiele

Nachdem in Abschnitt II.2 nachgewiesen wurde, daß jedes nebenläufige System (im Sinne dieser Arbeit) als Komposition sequentieller Komponenten aufgefaßt werden kann, wollen wir in diesem Abschnitt diese Idee nocheinmal von einem etwas anderen Standpunkt aus formulieren.

Unter einer sequentiellen Komponente verstehen wir ein System, das durch eine Menge lokaler Zustände und eine Menge lokaler Systemaktionen beschrieben wird. Diese Aktionen dürfen nicht nebenläufig stattfinden. Wenn sich ein nebenläufiges System (gegeben durch ein Spursystem $\mathcal{T}$) als Komposition von $K$ derartiger sequentieller Komponenten ergibt, so nennen wir jedes $i$ mit $1 \leqslant i \leqslant K$ einen *Agenten*. Die lokalen Aktionen des $i$-ten Agenten bezeichnen wir mit $A_i$, während seine lokalen Zustände (Positionen) mit $\Theta_i$ benannt werden; $\Theta_i$ ist hierbei eine Komponente in $\mathcal{T}$.

Formalisieren wir diese Idee: Wir beginnen mit dem Begriff des *verteilten Alphabets*; das ist eine Familie lokaler Alphabete $A_i$ für jeden Agenten $i$.

II.3.1. DEFINITION (Verteiltes Alphabet). Sei $J$ eine endliche nicht leere Menge von *Agenten* und sei $\boldsymbol{A} = \{A_i\}_{i\in J}$ eine Familie von Alphabeten; wir bezeichnen $\boldsymbol{A}$ als *verteiltes Alphabet*. Jedem verteilten Alphabet können wir ein Spuralphabet $\Sigma(\boldsymbol{A}) =_{\mathrm{Def}} \langle A, \bar{D}\rangle$ mit

$$A =_{\mathrm{Def}} \bigcup_{i\in J} A_i$$

und

$$a\ D\ b \Leftrightarrow_{\mathrm{Def}} \exists i \in J\,(a, b \in A_i)$$

zuordnen. □

II.3.2. DEFINITION (Verteiltes System). Sei $\mathcal{T}$ ein Spursystem über $\Sigma$, $J$ eine endliche nicht leere Menge von Agenten, $\boldsymbol{\Theta} = \{\Theta_i\}_{i\in J}$ eine Familie endlicher nicht leerer Positionsmengen mit $\Theta = \bigcup_{i\in J}\Theta_i$ und $\theta : \mathcal{T} \to \Theta$ eine Positionszuweisung an $\mathcal{T}$, so daß $\mathcal{T}$ $\boldsymbol{\Theta}$-überdeckt ist. Dann nennen wir die Struktur $\mathcal{D} = \langle \mathcal{T}, \theta, \boldsymbol{\Theta}\rangle$ ein *verteiltes System* über $\Sigma$ und $J$.

Jedes verteilte System $\mathcal{D} = \langle \mathcal{T}, \theta, \boldsymbol{\Theta}\rangle$ definiert ein verteiltes Alphabet $\boldsymbol{A}(\mathcal{D}) = \{A_i\}_{i\in J}$: Wir setzen

$$a \in A_i \Leftrightarrow_{\mathrm{Def}} \dot{\theta}(a) \cap \Theta_i \neq \varnothing \qquad \text{(II.3.}\alpha\text{)}$$

für alle $i \in J$. Sei weiterhin $A =_{\mathrm{Def}} \bigcup_{i\in J} A_i$. Die als *Agentenzuweisung* bezeichnete Operation $\mathrm{loc}_{\mathcal{D}} : A \to \mathcal{P}(J)$ ist durch

$$\mathrm{loc}_{\mathcal{D}}(a) =_{\mathrm{Def}} \{i \in J : a \in A_i\}$$

definiert. □

Man beachte, daß zwar jedes verteilte Alphabet $\boldsymbol{A}$ ein Spuralphabet $\Sigma(\boldsymbol{A})$ induziert und weiterhin jedes verteilte System $\mathcal{D}$ über $\Sigma$ und $J$ ein verteiltes Alphabet $\boldsymbol{A}(\mathcal{D})$ bestimmt, die Annahme $\Sigma = \Sigma(\boldsymbol{A}(\mathcal{D}))$ jedoch i. Allg. falsch ist. Es gilt $A_{\Sigma(\boldsymbol{A}(\mathcal{D}))} = A_\Sigma$ wegen Def. II.1.1.(a), die Relationen $I_\Sigma$ und $I_{\Sigma(\boldsymbol{A}(\mathcal{D}))}$ differieren jedoch in der folgenden Weise:

$$\begin{aligned} a\ I_\Sigma\ b &\Rightarrow \dot{\theta}(a) \cap \dot{\theta}(b) = \varnothing && \text{(Def. II.1.1.(b))}\\ &\Rightarrow \mathrm{loc}_{\mathcal{D}}(a) \cap \mathrm{loc}_{\mathcal{D}}(b) = \varnothing && \text{(Festlegung (II.3.}\alpha\text{))}\\ &\Rightarrow a\ I_{\Sigma(\boldsymbol{A}(\mathcal{D}))}\ b; && \text{(Def. } I_{\Sigma(\boldsymbol{A})}\text{)} \end{aligned}$$

oder äquivalent: $D_{\Sigma(\boldsymbol{A}(\mathcal{D}))} \subseteq D_\Sigma$.

Wir können diesen Umstand auch so interpretieren, daß wir ein verteiltes System $\mathcal{D}$ als Implementierung eines verteilten Alphabets $\boldsymbol{A}$ auffassen. $\mathcal{D}$ beachtet alle durch $\boldsymbol{A}$ spezifizierten Abhängigkeiten und enthält für jede in $\boldsymbol{A}$ vorkommende Aktion eine Implementierung, jedoch kann $\mathcal{D}$ weitere Abhängigkeiten enthalten, die nicht spezifiziert wurden.

Wir zeigen, daß Aktionen $a, b \in A_i$ für ein $i \in J$ tatsächlich nicht unabhängig sind:

II.3.3. THEOREM. *Sei $\mathcal{D}$ ein verteiltes System über $\Sigma$ und $J$ und sei $i \in J$. Dann gilt $a\ D_\Sigma\ b$ für alle $a, b \in A_i$.*

BEWEIS. $a, b \in A_i$ impliziert $a\ D_{\Sigma(\boldsymbol{A}(\mathcal{D}))}\ b$ und dies wiederum nach den obigen Ausführungen $a\ D_\Sigma\ b$. □

II.3.4. DEFINITION. Sei $\mathcal{D} = \langle \mathcal{T}, \theta, \{\Theta_i\}_{i\in J}\rangle$ ein verteiltes System über $\Sigma$ und $J$. Wir verwenden für den Rest dieser Arbeit die Festlegung

$$\boldsymbol{\Theta}_{\mathcal{D}} =_{\mathrm{Def}} \bigcup_{i\in J} \Theta_i.$$

□

Weiterhin wenden wir alle Begriffe, die für das Spursystem $\mathcal{T}_{\mathcal{D}}$ definiert sind, ebenso auf $\mathcal{D}$ an: Zum Beispiel nennen wir $\mathcal{D}$ reduziert, wenn $\mathcal{T}_{\mathcal{D}}$ reduziert ist, bezeichnen $\mathbf{LSSL}(\mathcal{T}_{\mathcal{D}})$ kurz mit $\mathbf{LSSL}(\mathcal{D})$, und $\overset{x}{\underset{\mathcal{D}}{\Longrightarrow}}$ steht für $\overset{x}{\underset{\mathcal{T}_{\mathcal{D}}}{\Longrightarrow}}$. In diesem Zusammenhang sei auch auf die in Abschnitt I.1 eingeführte Konvention zur Vermeidung „Indextreppen“ verwiesen: Die Komponenten $S_{\mathcal{T}_{\mathcal{D}}}$, $\delta_{\mathcal{T}_{\mathcal{D}}}$, ... von $\mathcal{D}$ werden kurz als $S_{\mathcal{D}}$, $\delta_{\mathcal{D}}$, ... geschrieben.

Die folgenden Beispiele zeigen, daß spezifische Formalismen als verteilte Systeme über einer Agentenmenge $J$ aufgefaßt werden können:

**Sichere Netzsysteme — 1-Invarianten.** Eine prominente Analysetechnik für Petri-Netze ist die Auswertung von Platzinvarianten. Ist $\mathcal{N}$ ein kontaktfreies Netzsystem, so nennen wir eine Abbildung $X : P_{\mathcal{N}} \to \mathbb{Z}$ eine Platzinvariante, wenn $X$ die Eigenschaft

$$\forall t \in T_{\mathcal{N}} \left( \sum_{p \in {}^\bullet t} X(p) = \sum_{p \in t^\bullet} X(p) \right)$$

erfüllt. Weiterhin setzen wir $X(p) \neq 0$ für wenigstens ein $p \in P_{\mathcal{N}}$ voraus. Für alle Platzinvarianten $X$ gilt das sog. *Markenerhaltungsgesetz*:

$$\forall Q \in S_{\mathcal{T}(\mathcal{N})} \left( \sum_{p \in P_{\mathcal{N}}} \vec{Q}_{\mathcal{N}}(p) \cdot X(p) = \sum_{p \in P_{\mathcal{N}}} \vec{Q}(p) \cdot X(p) \right), \qquad \text{(II.3.}\beta\text{)}$$

wobei $\vec{Q} : P_{\mathcal{N}} \to \{0,1\}$ die *charakteristische Funktion* einer Menge $Q \subseteq P_{\mathcal{N}}$ notiert, d. h.

$$\vec{Q}(p) =_{\text{Def}} \begin{cases} 1, & \text{falls } p \in Q; \\ 0, & \text{sonst.} \end{cases}$$

Eine spezielle Art von Platzinvariante ist eine 1-*Invariante*, d. h. eine Platzinvariante $X$, die $0 \leqslant X(p) \leqslant 1$ für alle $p \in P_{\mathcal{N}}$ sowie

$$\sum_{p \in P_{\mathcal{N}}} \vec{Q}_{\mathcal{N}}(p) \cdot X(p) = 1 \qquad \text{(II.3.}\gamma\text{)}$$

erfüllt. Sei $\boldsymbol{X}$ eine Menge von 1-Invarianten. Wir nennen $\mathcal{N}$ $\boldsymbol{X}$*-überdeckt*, wenn für alle $p \in P_{\mathcal{N}}$ eine 1-Invariante $X \in \boldsymbol{X}$ mit $X(p) > 0$ existiert.

Aus (II.3.$\beta$) und (II.3.$\gamma$) folgt unmittelbar: Wenn wir für eine 1-Invariante $X$

$$\Theta_X =_{\text{Def}} \{p \in P_{\mathcal{N}} : X(p) > 0\}$$

und für eine Menge $\boldsymbol{X}$ von 1-Invarianten

$$\boldsymbol{\Theta_X} =_{\text{Def}} \{\Theta_X\}_{X \in \boldsymbol{X}}$$

setzen, so ist $\mathcal{T}(\mathcal{N})$ $\boldsymbol{\Theta_X}$-überdeckt, wenn $\mathcal{N}$ $\boldsymbol{X}$-überdeckt ist (man beachte, daß es nur endlich viele (allerdings exponentiell viele) Abbildungen $X : P_{\mathcal{N}} \to \{0,1\}$ gibt).

II.3.5. Bemerkung. Zur Berechnung von Platzinvarianten wird zunächst die *Inzidenzmatrix* eines Netzsystems $\mathcal{N}$ gebildet. Nehmen wir

$$P_{\mathcal{N}} = \{p_1, p_2, \ldots, p_n\} \text{ und } T_{\mathcal{N}} = \{t_1, t_2, \ldots, t_k\}$$

an. Die Inzidenzmatrix von $\mathcal{N}$ ist eine $n \times k$-Matrix $\boldsymbol{C} = (c_{i,j})$ mit

$$c_{i,j} =_{\text{Def}} \begin{cases} 1, & p_i \in t_j^{\cdot} - {}^{\cdot}t_j; \\ -1, & p_i \in {}^{\cdot}t_j - t_j^{\cdot}; \\ 0, & \text{sonst.} \end{cases}$$

Dann entspricht jede nichttriviale Lösung $\tilde{X}$ des homogenen linearen Gleichungssystem

$$\tilde{X} \cdot \boldsymbol{C} = \boldsymbol{0}$$

einer Platzinvariante. Dabei bezeichnen wir $\tilde{X}$ als *nichttrivial*, wenn es wenigstens eine Komponente $x_i$ von $\tilde{X}$ mit $x_i \neq 0$ gibt. „Entspricht“ bedeutet: Bei $\tilde{X} = \langle x_1, x_2, \ldots, x_n \rangle$ haben wir es mit einem Vektor zu tun, weiter oben haben wir Platzinvarianten jedoch als Abbildungen $X : P_{\mathcal{N}} \to \mathbb{Z}$ definiert. Diese formale Unstimmigkeit wird jedoch durch die Festlegung $X(p_i) =_{\text{Def}} x_i$ für $1 \leqslant i \leqslant n$ beseitigt. □

Die Literatur zu Invarianten in Petri-Netzen ist reichhaltig; der interessierte Leser sei etwa auf [**19, 20, 76**] verwiesen.

**II.3.1. Kommunizierende endliche sequentielle Prozesse.** Neben der im vorhergehenden Abschnitt diskutierten analytischen Methode erlauben viele Darstellungsformen für nebenläufige Systeme eine Herleitung einer Komponentenüberdeckung aus ihrer syntaktischen Struktur. Ein einfaches Beispiel die Komposition endlicher Zustandsmaschinen. Eine Zustandsmaschine ist ein Transitionssystem über einem Alphabet $A$. Die Kommunikation solcher Zustandsmaschinen folgt dem CSP-Modell. Zwei oder mehrere solcher Zustandsmaschinen kommunizieren mit Hilfe gemeinsamer Aktionen, indem sie diese gemeinsam ausführen. Dabei kann das komponierte System nur dann eine Aktion ausführen, wenn alle Komponenten, deren Aktionsalphabet diese Aktion enthält, diese Aktion ausführen können.

Seien $\mathcal{T}_i$ für $i = 1, 2, \ldots, n$ und $n > 1$ endliche deterministische Transitionssysteme über den Alphabeten $A_i$, wobei wir $S_{\mathcal{T}_i} \cap S_{\mathcal{T}_j} = \varnothing$ für $1 \leqslant i \neq j \leqslant n$ annehmen wollen. Setzen wir $A =_{\text{Def}} \bigcup_{i=1}^{n} A_i$ und $\Sigma =_{\text{Def}} \left\langle A, \bar{D} \right\rangle$, wobei

$$a \; D \; b =_{\text{Def}} \exists i \, (1 \leqslant i \leqslant n \;\&\; a, b \in A_i)$$

gilt. Offenbar ist $\Sigma$ ein Spuralphabet.

Die *Komposition* $\mathcal{T} = \|_{i=1}^{n} \mathcal{T}_i$ ist wie folgt definiert:

(a) $S_{\mathcal{T}} =_{\text{Def}} S_{\mathcal{T}_1} \times S_{\mathcal{T}_2} \times \cdots \times S_{\mathcal{T}_n}$

(b) $s_{\mathcal{T}} =_{\text{Def}} \langle s_{\mathcal{T}_1}, s_{\mathcal{T}_2}, \ldots, s_{\mathcal{T}_n} \rangle$,

(c) $\delta_{\mathcal{T}}(\langle s_1, s_2, \ldots, s_n \rangle, a) =_{\text{Def}} \langle s_1', s_2', \ldots, s_n' \rangle$ ist definiert, wenn für alle $i$ mit $1 \leqslant i \leqslant n$ die Implikation

$$a \in A_i \Rightarrow \delta_{\mathcal{T}_i}(s_i, a) \text{ definiert}$$

erfüllt ist und weiterhin $\delta_{\mathcal{T}_i}(s_i, a)$ für wenigstens eines dieser $i$ definiert ist. In diesem Fall gilt

$$s_i' = \begin{cases} \delta_{\mathcal{T}_i}(s_i, a), & \text{falls } a \in A_i; \\ s_i, & \text{sonst.} \end{cases} \qquad (\text{für } 1 \leqslant i \leqslant n)$$

Natürlich ist $\mathcal{T}$ ein Transitionssystem, wesentlicher ist jedoch, daß es sich bei $\mathcal{T}$ um ein Spursystem über $\Sigma$ handelt. Wir schreiben $\mathcal{T}_1 \parallel \mathcal{T}_2$, falls $n = 2$ ist.

Unsere Definition ist abhängig von der Nummerierung der Transitionssysteme $\mathcal{T}_i$, allerdings ist leicht zu sehen, daß

(a) $\mathcal{T}_1 \parallel \mathcal{T}_2$ isomorph zu $\mathcal{T}_2 \parallel \mathcal{T}_1$ und
(b) $\mathcal{T}_1 \parallel (\mathcal{T}_2 \parallel \mathcal{T}_3)$ isomorph zu $(\mathcal{T}_1 \parallel \mathcal{T}_2) \parallel \mathcal{T}_3$

ist.

Eine Positionszuweisung $\theta : \|_{i=1}^n \mathcal{T}_i \to \bigcup_{i=1}^n S_{\mathcal{T}_i}$ ist nun leicht zu finden:

(a) $\theta(\langle s_1, s_2, \ldots, s_n\rangle) =_{\text{Def}} \{s_1, s_2, \ldots, s_n\}$,
(b) $s \in \dot{\theta}(a) \Leftrightarrow_{\text{Def}} \exists i \left(1 \leqslant i \leqslant n \;\&\; s \xrightarrow[\mathcal{T}_i]{a}\right)$,
(c) $s \in \theta^{\cdot}(a) \Leftrightarrow_{\text{Def}} \exists i \left(1 \leqslant i \leqslant n \;\&\; \exists s' \in S_{\mathcal{T}_i} \left(s' \xrightarrow[\mathcal{T}_i]{a} s\right)\right)$.

Dann ist $\theta$ kontaktfrei, kausal- und konfliktvollständig. Wählen wir weiterhin $\boldsymbol{\Theta} =_{\text{Def}} \{S_{\mathcal{T}_i}\}_{i=1,2,\ldots,n}$, so ist $\|_{i=1}^n \mathcal{T}_i$ $\boldsymbol{\Theta}$-überdeckt.

**II.3.2. Registersysteme.** Nachdem wir uns im vorherigen Abschnitt mit einem synchronen Kommunikationsmodell für endliche Zustandsmaschinen befaßt haben, wenden wir uns nun einer als „Registersystem“ bezeichneten Darstellungsform für nebenläufige Systeme zu, der ein asynchrones Kommunikationsmodell zugrundeliegt. Registernetze wurden in [**25**] zur Analyse von Steuerungssoftware eingeführt. Für eine ausführliche Diskussion dieser Netzklasse, Beispiele usw. sei auf [**25**] verwiesen.

Für diesem Abschnitt führen wir einige weitere Schreibweisen ein:

II.3.6. Definition. $A^n$ bezeichnet die Menge

$$\underbrace{A \times A \times \cdots \times A}_{n\text{-mal}}$$

Für $i = 1, 2, \ldots, n$ ist die Abbildung $\text{pr}_i : A_1 \times A_2 \times \cdots \times A_n \to A_i$ als

$$\text{pr}_i(\langle a_1, a_2, \ldots, a_n\rangle) =_{\text{Def}} a_i$$

definiert. $\text{pr}_i$ heißt die *Projektion* von $\langle a_1, a_2, \ldots, a_n\rangle$ auf die $i$-te Komponente.

Wenn $f : A^n \rightharpoonup B^m$ eine partielle Abbildung und $\perp$ ein ausgezeichnetes Element mit $\perp \notin A$ und $\perp \notin B$ ist, verwenden wir die Notation $f^{\perp} : A_{\perp}^n \to B_{\perp}^m$, um die

(totale) Funktion

$$f^{\perp}(a_1, a_2, \ldots, a_n) =_{\text{Def}} \begin{cases} \perp^m, \text{falls } a_i = \perp \text{ für ein } i \text{ mit } 1 \leqslant i \leqslant n \\ \quad \text{oder } f(a_1, a_2, \ldots, a_n) \text{ undefiniert ist;} \\ f(a_1, a_2, \ldots, a_n) \text{ sonst} \end{cases}$$

zu bezeichnen, wobei $A_{\perp} =_{\text{Def}} A \cup \{\perp\}$ für jede Menge $A$ mit $\perp \notin A$ und $\perp^m \in B_{\perp}^m$ derjenige Vektor mit $\text{pr}_j(\perp^m) = \perp$ für alle $j$ mit $1 \leqslant j \leqslant m$ ist. □

II.3.7. Definition (Registernetz). Ein Registernetz ist eine Struktur $\mathscr{R} = \langle P, T, F, V, \text{sig}, \boldsymbol{G}, \boldsymbol{P}, s\rangle$ mit den folgenden Komponenten:

(a) Einer Menge von *Plätzen* $P$, einer Menge von *Transitionen* $T$ und einer *Flußrelation* $F \subseteq (P \times T) \cup (T \times P)$, so daß $\langle P, T, F\rangle$ ein Netz ist.

(b) Einer endlichen (potentiell leeren) Menge $V$ von *Registern* mit $V \cap T = \varnothing = V \cap P$.

(c) Einer Abbildung $\text{sig} : T \to V^* \times V^*$. $\text{sig}(t)$ heißt *Signatur* von $T$. Um den Schreibaufwand zu reduzieren, verwenden wir die folgenden abkürzenden Schreibweisen. Sei $\text{sig}(t) = \langle v_1 v_2 \ldots v_n, w_1 w_2 \ldots w_k\rangle$:

(i) $\text{ar}^-(t) =_{\text{Def}} n$ und $\text{ar}^+(t) =_{\text{Def}} k$. $\text{ar}^-(t)$ heißt *Eingabestelligkeit* von $t$, $\text{ar}^+(t)$ wird als *Ausgabestelligkeit* von $t$ bezeichnet.

(ii) $\text{in}(t, i) =_{\text{Def}} v_i$ für $1 \leqslant i \leqslant \text{ar}^-(t)$ und $\text{out}(t, j) =_{\text{Def}} w_j$ für $1 \leqslant j \leqslant \text{ar}^+(t)$. in und out heißen *Eingabe-* bzw. *Ausgaberegisterselektoren.*

(iii) $\text{mod}(t) =_{\text{Def}} \{\text{out}(t, i) : 1 \leqslant j \leqslant \text{ar}^+(t)\}$ bezeichnet die Menge der durch $t$ *modifizierten Register.*

(iv) $\text{acc}(t) =_{\text{Def}} \{\text{in}(t, i) : 1 \leqslant i \leqslant \text{ar}^-(t)\} \cup \text{mod}(t)$ ist die Menge der Register, auf die $t$ *zugreift.*

Wir nehmen an, daß für alle $t \in T$ und für $1 \leqslant i, j \leqslant \text{ar}^-(t)$

$$\text{in}(t, i) = \text{in}(t, j) \Rightarrow i = j$$

gilt, ebenso wie wir

$$\text{out}(t, i) = \text{out}(t, j) \Rightarrow i = j$$

für $t \in T$, $1 \leqslant i, j \leqslant \text{ar}^+(t)$ voraussetzen.

(d) $\boldsymbol{P} = \{P_t\}_{t \in T}$ ist eine Familie von partiellen Abbildungen $P_t : \mathbb{N}^{\text{ar}^-(t)} \rightharpoonup \mathbb{B}$. $P_t$ heißt *Prädikat (eines Registernetzes)* von $t$.

(e) $\boldsymbol{G} = \{G_t\}_{t \in T}$ ist eine Familie von paritiellen Abbildungen $G_t : \mathbb{N}^{\text{ar}^-(t)} \rightharpoonup \mathbb{N}^{\text{ar}^+(t)}$. $G_t$ wird als *Schaltfunktion (eines Registernetzes)* zu $t$ bezeichnet. Für den Fall $\text{ar}^+(t) = 0$ sei $G_t =_{\text{Def}} \text{id}_{\mathbb{N}}$, diese Wahl ist willkürlich, jede andere Abbildung auf $\mathbb{N}$ hätte ebenso verwendet werden können.

(f) $s \in \mathscr{P}(P) \times (V \to \mathbb{N})$ heißt *initialer Zustand* von $\mathscr{R}$.

□

II.3.8. DEFINITION (Zustand). Ist $\mathscr{R}$ ein Registernetz, so bezeichnen wir jede Menge $Q \subseteq P_{\mathscr{R}}$ als *Markierung* von $\mathscr{R}$. Eine Abbildung $\eta : V_{\mathscr{R}} \to \mathbb{N}_\perp$ heißt *Registerzustand* von $\mathscr{R}$. Ist $Q$ eine Markierung und $\eta$ ein Registerzustand von $\mathscr{R}$, so heißt das Paar $s = \langle Q, \eta \rangle$ *Zustand* von $\mathscr{R}$.

Ein Registerzustand $\eta$ heißt $K$-beschränkt für eine Konstante $K \in \mathbb{N}$, wenn $\eta(v) \neq \perp \Rightarrow \eta(v) < K$ gilt. $\eta$ heißt *beschränkt*, wenn es ein $K > 0$ gibt, so daß $\eta$ $K$-beschränkt ist. Ein Zustand $s$ von $\mathscr{R}$ heißt *$K$-beschränkt* bzw. *beschränkt*, wenn $\eta_s$ $K$-beschränkt bzw. beschränkt ist. □

II.3.9. DEFINITION. Ein Registernetz $\mathscr{R}$ heißt *$K$-beschränkt* für eine Konstante $K > 0$, wenn der Zustand $s_{\mathscr{R}}$ $K$-beschränkt ist und für alle $t \in T_{\mathscr{R}}$ und für alle $n_1, n_2, \ldots, n_{\mathrm{ar}^-(t)} \in \mathbb{N}$ die Implikation

$$\begin{aligned} n_i < K \;\&\; G_t^\perp\left(n_1, n_2, \ldots, n_{\mathrm{ar}^-(t)}\right) &\neq \perp^{\mathrm{ar}^+(t)} \\ \Rightarrow \quad \mathrm{pr}_j\left(G_t^\perp\left(n_1, n_2, \ldots, n_{\mathrm{ar}^-(t)}\right)\right) &< K \end{aligned}$$

erfüllt ist, wobei $1 \leqslant i \leqslant \mathrm{ar}^-(t)$ und $1 \leqslant j \leqslant \mathrm{ar}^+(t)$ gilt. $\mathscr{R}$ heißt *beschränkt* wenn $\mathscr{R}$ $K$-beschränkt für ein $K > 0$ ist. □

II.3.10. DEFINITION. Sei $\mathscr{R}$ ein Registernetz. Wir definieren eine *Transitionsfunktion*

$$\delta_{\mathscr{R}} : (\mathscr{P}(Q_{\mathscr{R}}) \times (V_{\mathscr{R}} \to \mathbb{N}_\perp)) \times T_{\mathscr{R}} \rightharpoonup (\mathscr{P}(Q_{\mathscr{R}}) \times (V_{\mathscr{R}} \to \mathbb{N}_\perp))$$

für $\mathscr{R}$ durch $\delta_{\mathscr{R}}(s_1, t) =_{\mathrm{Def}} s_2$ für $s_1 = \langle Q_1, \eta_1 \rangle, s_2 = \langle Q_2, \eta_2 \rangle$, wobei

$$Q_1 \subseteq {}^\cdot t \;\&\; Q_1 \cap (t^\cdot - {}^\cdot t) = \varnothing \;\&\; Q_2 = (Q_1 - {}^\cdot t) \cap t^\cdot \qquad \text{(II.3.}\delta\text{)}$$

sowie

$$\begin{aligned} &P_t^\perp(\eta_1(\mathrm{in}(t,1)), \eta_1(\mathrm{in}(t,2)), \ldots, \eta_1(\mathrm{in}(t,\mathrm{ar}^-(t)))) \neq \mathrm{false} \\ &\;\&\; \eta_2(v) = \begin{cases} \mathrm{pr}_j(G_t^\perp(\eta_1(\mathrm{in}(t,1)), \eta_1(\mathrm{in}(t,2)), \ldots, \eta_1(\mathrm{in}(t,\mathrm{ar}^-(t))))), \\ \qquad \text{falls } 1 \leqslant j \leqslant \mathrm{ar}^+(t) \;\&\; v = \mathrm{out}(t,j); \\ \eta_1(v), \text{sonst} \end{cases} \end{aligned} \qquad \text{(II.3.}\epsilon\text{)}$$

gilt. Ist $\delta_{\mathscr{R}}(s_1, t)$ definiert, notieren wir dies durch $s_1 \xrightarrow[\mathscr{R}]{t}$, gilt $s_1 \xrightarrow[\mathscr{R}]{t}$ und $\delta_{\mathscr{T}}(s_1, t) = s_2$, so schreiben wir auch $s_1 \xrightarrow[\mathscr{R}]{t} s_2$. □

Bedingung (II.3.$\delta$) ist einfach die Definition der Schaltregel für Netzsysteme (vgl. Beispiel I.3.5). (II.3.$\epsilon$) definiert eine Transition $t$ als konzessioniert, wenn das ihr zugeordnete Prädikat $P_t^\perp$ angewendet auf die Werte der Eingaberegister von $t$ entweder zu dem Wert $\perp$ oder zu dem Wert true evaluiert. Eine ausführliche Diskussion, die die Sinnfälligkeit diese Definition zumindest für verteilte Steuerungen begründet, findet sich in [**25**]. Wenn $t$ schaltet, erhalten ihre Ausgaberegister neue Werte entsprechend der Schaltfunktion $G_t^\perp$ angewendet auf die Werte der Eingaberegister von $t$; ist $v = \mathrm{out}(t,j)$ eines dieser Ausgaberegister, so erhält $v$ als

neuen Wert die $j$-te Projektion des Ergebnisvektors von $G_t^\perp$. Register, die nicht als modifizierte Register $v' \in \mathrm{mod}(t)$ vorkommen, behalten ihre urspünglichen Werte.

II.3.11. DEFINITION. Ist $\mathscr{R}$ ein Registernetz, so ist die Relation $I_{\mathscr{R}} \subseteq T_{\mathscr{R}} \times T_{\mathscr{R}}$ als

$$\begin{aligned} t_1 \; I_{\mathscr{R}} \; t_2 \quad \Leftrightarrow_{\mathrm{Def}} \quad & \dot{t}_1 \cap \dot{t}_2 = \varnothing \\ & \& \; \mathrm{mod}(t_1) \cap \mathrm{acc}(t_2) = \varnothing \; \& \; \mathrm{mod}(t_2) \cap \mathrm{acc}(t_1) = \varnothing \end{aligned}$$

definiert. Offenbar ist $I_{\mathscr{R}}$ eine Unabhängigkeitsrelation und damit $\Sigma_{\mathscr{R}} =_{\mathrm{Def}} \langle T_{\mathscr{R}}, I_{\mathscr{R}} \rangle$ ein Spuralphabet. □

II.3.12. DEFINITION. Wir bestimmen ein Transitionssystem $\mathscr{T}(\mathscr{R})$ über $T_{\mathscr{R}}$ für ein Registernetz $\mathscr{R}$, indem wir die Komponenten von $\mathscr{T}(\mathscr{R})$ wie folgt definieren:

(a) $S_{\mathscr{T}(\mathscr{R})}$ ist die kleinste Menge, die

$$s_{\mathscr{R}} \in S_{\mathscr{T}(\mathscr{R})} \; \& \; \forall t \in T_{\mathscr{R}} \left( \bigcup_{s \in S_{\mathscr{T}(\mathscr{R})}} \delta_{\mathscr{R}}(s,t) \subseteq S_{\mathscr{T}(\mathscr{R})} \right)$$

erfüllt.

(b) $s_{\mathscr{T}(\mathscr{R})} =_{\mathrm{Def}} s_{\mathscr{R}}$.

(c) $\delta_{\mathscr{T}(\mathscr{R})}(s,t) =_{\mathrm{Def}} \delta_{\mathscr{R}}(s,t)$ für alle $t \in T_{\mathscr{R}}$, $s \in S_{\mathscr{T}(\mathscr{R})}$.

□

II.3.13. LEMMA. *Für jedes beschränkte Registernetz $\mathscr{R}$ ist $\mathscr{T}(\mathscr{R})$ ein endliches, reduziertes und deterministisches Spursystem über $\Sigma_{\mathscr{R}}$.*

BEWEIS. Die Endlichkeit von $\mathscr{T}(\mathscr{R})$ folgt mit Hilfe einer einfachen Induktion über der Länge einer Schaltfolge $\sigma \in T_{\mathscr{R}}^*$: Man zeigt, daß alle erreichbaren Zustände $s \in S_{\mathscr{R}}$ $K$-beschränkt sind. Dann folgert man, daß es nur endlich viele $K$-beschränkte Zustände von $\mathscr{R}$ geben kann.

$\mathscr{T}(\mathscr{R})$ ist nach Definition reduziert. $\mathscr{T}(\mathscr{R})$ ist deterministisch, da $G_t$ eine partielle Funktion für alle $t \in T_{\mathscr{R}}$ ist. Damit stehen lediglich die Rauteneigenschaften I.3.12, (a) bis (c)) in Frage.

Wir demonstrieren lediglich I.3.12.(a), die beiden anderen Eigenschaften folgen analog.

Mit $\mathrm{acc}(t_1) \cap \mathrm{mod}(t_2) = \varnothing$ gilt $\eta_1(v) = \eta_3(v)$ für alle $v \in \mathrm{acc}(t_1)$, damit folgt weiter

$$\begin{aligned} & P_{t_1}^\perp \left( \eta_1(v_1), \eta_1(v_2), \ldots, \eta_1(v_{\mathrm{ar}^-(t_1)}) \right) \neq \mathrm{false} \\ \Leftrightarrow \quad & P_{t_1}^\perp \left( \eta_3(v_1), \eta_3(v_2), \ldots, \eta_3(v_{\mathrm{ar}^-(t_1)}) \right) \neq \mathrm{false}. \end{aligned}$$

Sei $\mathcal{N} = \langle P_{\mathscr{R}}, T_{\mathscr{R}}, F_{\mathscr{R}}, Q_{s_{\mathscr{R}}} \rangle$ das $\mathscr{R}$ zugrundeliegende Netzsystem. Offenbar gilt

$$S_{\mathcal{N}} \supseteq \{ Q \subseteq P_{\mathscr{R}} : \exists \eta \in (V_{\mathscr{R}} \to \mathbb{N}_\perp) \, (\langle Q, \eta \rangle \in S_{\mathscr{R}}) \}$$

nach Definition von $\mathcal{T}(\mathcal{N})$ (vgl. Beispiel I.3.5). Sind $s_1, s_2, s_3 \in S_{\mathcal{T}(\mathcal{R})}$ Zustände mit $s_i = \langle Q_i, \eta_i \rangle$ für $i = 1, 2, 3$ und $t_1, t_2 \in T_{\mathcal{R}}$ Transitionen von $\mathcal{R}$ mit $t_1 \; I_{\mathcal{R}} \; t_2$, so daß $s_1 \xrightarrow[\mathcal{R}]{t_1} s_2$ und $s_1 \xrightarrow[\mathcal{R}]{t_2} s_3$ gilt, gibt es eine Menge $Q_4 \subseteq P_{\mathcal{R}}$ mit $Q_2 \xrightarrow[\mathcal{N}]{t_2} Q_4$ und $Q_3 \xrightarrow[\mathcal{N}]{t_1} Q_4$; die Argumentation aus Beispiel I.3.15 kann hier angewendet werden.

Schließlich stellen wir wegen $\mathrm{acc}(t_1) \cap \mathrm{mod}(t_2) = \varnothing$

$$G^{\perp}_{t_1}(\eta_1(v_1), \eta_1(v_2), \ldots, \eta_1(v_{\mathrm{ar}^-(t_1)})) = G^{\perp}_{t_1}(\eta_3(v_1), \eta_3(v_2), \ldots, \eta_3(v_{\mathrm{ar}^-(t_1)}))$$

fest. Wiederholen wir diese Argumentation für die Zustandsübergänge $s_1 \xrightarrow[\mathcal{R}]{t_2} s_3$ und $s_2 \xrightarrow[\mathcal{R}]{t_2}$, erhalten wir einen Registerzustand $\eta_4$, der zusammen mit $Q_4$ den gewünschten Zustand $s_4 = \langle Q_4, \eta_4 \rangle$ bildet. □

II.3.14. Definition (Zustandsmaschine). Ein Netzsystem $\mathcal{N}$ wird als *Zustandsmaschine* bezeichnet, wenn $|{}^{\cdot}t| \leqslant 1$ und $|t^{\cdot}| \leqslant 1$ für alle Transitionen $t \in T_{\mathcal{N}}$ und weiterhin $|Q_{\mathcal{N}}| \leqslant 1$ gilt. □

II.3.15. Definition (Registersystem). Seien $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_n$ Zustandsmaschinen, so daß $P_{\mathcal{N}_i}$ und $T_{\mathcal{N}_j}$ für $1 \leqslant i \neq j \leqslant n$ paarweise disjunkt sind. Ein Registernetz $\mathcal{R}$ heißt *Registersystem*, wenn

(a) $P_{\mathcal{R}} = \bigcup_{i=1}^{n} P_{\mathcal{N}_i}$, $T_{\mathcal{R}} = \bigcup_{i=1}^{n} T_{\mathcal{N}_i}$, $F_{\mathcal{R}} = \bigcup_{i=1}^{n} F_{\mathcal{N}_i}$ sowie
(b) $Q_{\mathcal{R}} =_{\mathrm{Def}} \bigcup_{i=1}^{n} Q_{\mathcal{N}_i}$ gilt, wobei $s_{\mathcal{R}} = \langle Q_{\mathcal{R}}, \eta_{\mathcal{R}} \rangle$ sei.

□

Wiederum ist eine Positionszuweisung für ein Registersystem $\mathcal{R}$ leicht zu finden: $\theta(\langle Q, \eta \rangle) =_{\mathrm{Def}} Q$ für alle Zustände $\langle Q, \eta \rangle$ von $\mathcal{R}$, ${}^{\cdot}\theta(t) =_{\mathrm{Def}} {}^{\cdot}t$ und $\theta^{\cdot}(t) =_{\mathrm{Def}} t^{\cdot}$ für alle $t \in T_{\mathcal{R}}$. Aus der Petri-Netz-Theorie ist bekannt, daß für Zustandsmaschinen $\mathcal{N}$ die Implikation

$$|Q_{\mathcal{N}}| \leqslant 1 \Rightarrow \forall Q \in S_{\mathcal{T}(\mathcal{N})}\,(|Q| \leqslant 1)\,,$$

gilt, die keineswegs überraschende Wahl $\Theta = \{P_{\mathcal{N}_i}\}_{i=1,2,\ldots,n}$ als Familie von Komponenten zur Überdeckung von $\mathcal{R}$ ist also korrekt.

Man beachte, daß $\theta$ weder konflikt- noch kausalvollständig ist, da Konflikte und Kausalbeziehungen, die sich aufgrund des gemeinsamen Zugriffs zweier Transitionen auf dasselbe Register ergeben, nicht berücksichtigt werden.

KAPITEL III

# Prozeßautomaten

Dieses Kapitel behandelt Prozeßautomaten. Die Definition ihrer Struktur und der durch sie erkannten Sprache findet sich in Abschnitt III.1. Die Begriffe der Verzweigungs-, Vereinigungs- und Rekurrenzvollständigkeit werden behandelt, die als Minimalitätskriterien für Prozeßautomaten verwendet werden.

In Abschnitt III.2 stellen wir einen Grundalgorithmus vor, der in III.3 auf verschiedene Arten verbessert wird. Wir weisen nach, daß beide Algorithmen vollständige Prozeßautomaten im Sinne von Abschnitt III.1 erzeugen, die jedoch i. d. R. nicht minimal sind.

## III.1. Motivation und Definition

Ein Prozeßautomat besteht aus einer endlichen Menge von Zuständen, die einen ausgezeichneten Initialzustand enthält, und einer partiellen Transitionsfunktion. Transitionen finden jedoch nicht durch die Anwendung lediglich einzelner Systemaktionen statt, die Transitionsfunktion ist für Paare von Zuständen und Semiordnungen definiert.

Analog zu dem klassischen Automatenbegriff (der etwas abgewandelt in dieser Arbeit in den Transitionssystemen seine begriffliche Entsprechung findet), existiert auch für Prozeßautomaten ein Sprachbegriff: Sind die Semiordnungen, die die Transitionen eines Prozeßautomaten ausmachen, Elemente der Klasse $\mathbf{CSO}(\Sigma)$ für ein Spuralphabet $\Sigma$, so ergibt sich ein Semiwort der durch diesen Automaten akzeptierten Semisprache als die Verkettung hintereinander ausführbarer Semiordnungen mit Hilfe der $\circ_\Sigma$-Operation.

III.1.1. Definition. Ein *Prozeßautomat* über einem Spuralphabet $\Sigma$ ist eine Struktur $\mathcal{A} = \langle S, X, \delta, s\rangle$ mit den folgenden Komponenten:

(a) $S$ ist eine endliche Menge von *Zuständen*,
(b) $X \subseteq \mathbf{CSO}(\Sigma)$ ist eine endliche Menge von Semiordnungen,
(c) $\delta : S \times X \rightharpoonup S$ ist eine *Transitionsfunktion*,
(d) $s \in S$ ist ein *initialer Zustand*.

Analog zu den in Abschnitt I.3 eingeführten Schreibweisen für Transitionssysteme verwenden wir die „Pfeilnotation“

$$s_1 \xrightarrow[\mathcal{A}]{x} s_2 \Leftrightarrow_{\text{Def}} s_2 = \delta(s_1, x) \text{ definiert}$$

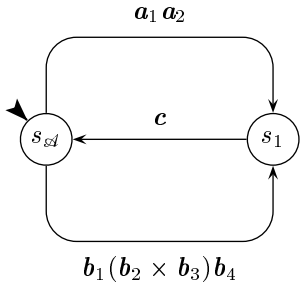



ABBILDUNG III.1. Ein Prozeßautomat.

und

$$s_1 \xrightarrow[\mathscr{A}]{x} \Leftrightarrow_{\text{Def}} \exists s_2 \in S \left( s_1 \xrightarrow[\mathscr{A}]{x} s_2 \right).$$

$\mathscr{A}$ heißt *endlich*, wenn $S$ endlich ist. □

III.1.2. BEISPIEL. Abbildung III.1 zeigt einen Prozeßautomaten. Seine Darstellung folgt der Darstellungsweise für Transitionssysteme: Kreise stellen die Zustände des Automaten dar, die Transitionsfunktion wird mit Hilfe von Pfeilen gezeichnet. Der Initialzustand ist durch einen einlaufenden Pfeil gekennzeichnet. □

III.1.3. DEFINITION. Sei $\mathscr{A}$ ein Prozeßautomat über $\Sigma$. Ein *Weg* durch $\mathscr{A}$, der von einem Zustand $s \in S_{\mathscr{A}}$ zu einem Zustand $s' \in S_{\mathscr{A}}$ führt, ist eine Folge von Semiordnungen $\chi = x_0 x_1 \ldots x_{n-1} \in X_{\mathscr{A}}^{*}$[1], so daß Zustände $s_0, s_1, \ldots, s_n \in S_{\mathscr{A}}$ mit $s = s_0$, $s' = s_n$ und $s_i \xrightarrow[\mathscr{A}]{x_i} s_{i+1}$ für $0 \leqslant i < n$ existieren. $\mathbf{P}_{\mathscr{A}}(s, s')$ bezeichnet die Menge aller Wege durch $\mathscr{A}$, die von $s$ nach $s'$ führen. Weiterhin setzen wir

$$\mathbf{P}_{\mathscr{A}}(s) =_{\text{Def}} \bigcup_{s' \in S_{\mathscr{A}}} \mathbf{P}_{\mathscr{A}}(s, s') \text{ sowie } \mathbf{P}(\mathscr{A}) =_{\text{Def}} \mathbf{P}_{\mathscr{A}}(s_{\mathscr{A}})$$

Ist $\chi = x_0 x_1 \ldots x_{n-1} \in \mathbf{P}_{\mathscr{A}}(s, s')$, so bezeichnen wir mit $\text{con}_{\Sigma}(\chi)$ die Semiordnung

$$x_0 \circ_{\Sigma} x_1 \circ_{\Sigma} \cdots \circ_{\Sigma} x_{n-1}.$$

Die *Semisprache* von $\mathscr{A}$ is

$$\mathbf{SL}(\mathscr{A}) =_{\text{Def}} \{\boldsymbol{x} \in \mathbf{CSW}(A_{\Sigma}) : \exists \chi \in \mathbf{P}(\mathscr{A})\,(x \leqslant \text{con}_{\Sigma}(\chi))\}$$

Sind $s, s' \in S_{\mathscr{A}}$ Zustände und $x \in \mathbf{CSO}(\Sigma)$, so schreiben wir $s \xRightarrow[\mathscr{A}]{x}$, wenn es einen Weg $\chi \in \mathbf{P}_{\mathscr{A}}(s, s')$ mit $x \leqslant \text{con}_{\Sigma}(\chi)$ gibt; gilt darüber hinaus $x \equiv \text{con}_{\Sigma}(\chi)$ so schreiben wir $s \xRightarrow[\mathscr{A}]{x} s'$. Es gilt also $\boldsymbol{x} \in \mathbf{SL}(\mathscr{A}) \Leftrightarrow s_{\mathscr{A}} \xRightarrow[\mathscr{A}]{x}$. □

[1]Dies ist nicht eindeutig, da $\chi$ ebenso $x_0 \cdot x_1 \cdot \ldots \cdot x_{n-1}$ bezeichnen könnte. Gemeint ist aber die Folge bestehend aus den Elementen $x_0, x_1, \ldots, x_{n-1}$.

III.1.4. DEFINITION. Sei $\mathscr{A}$ ein Prozeßautomat über $\Sigma$ und $\mathscr{T}$ ein Spursystem über demselben Spuralphabet. $\mathscr{A}$ heißt *Prozeßautomat* zu $\mathscr{T}$, wenn

(a) $s_{\mathscr{A}} = s_{\mathscr{T}}$,
(b) $S_{\mathscr{A}} \subseteq S_{\mathscr{T}}$, sowie
(c) $s \xrightarrow[\mathscr{A}]{x} s' \Rightarrow s \overset{x}{\underset{\mathscr{T}}{\Longrightarrow}} s'$ für alle $s, s' \in S_{\mathscr{A}}$ und $x \in X_{\mathscr{A}}$

gilt. □

III.1.5. BEISPIEL. Der Prozeßautomat aus Beispiel III.1.2 (S. 74) ist ein Prozeßautomat zu dem Spursystem aus Beispiel I.3.2 (S. 33). □

III.1.6. LEMMA. *Sei $\mathscr{A}$ ein Prozeßautomat zu dem Spursystem $\mathscr{T}$ über $\Sigma$. Dann gilt* $\mathbf{SL}(\mathscr{A}) \subseteq \mathbf{SL}(\mathscr{T})$.

BEWEIS. Das Lemma folgt mit Hilfe einer einfachen Induktion über der Länge von Wegen durch $\mathscr{A}$ unter Zurhilfenahme von Korollar I.3.43. □

III.1.7. DEFINITION. Sei $\mathscr{A}$ ein Prozeßautomat zu dem Spursystem $\mathscr{T}$ über $\Sigma$. $\mathscr{A}$ heißt *vollständig* bzgl. $\mathscr{T}$, wenn $\mathbf{SL}(\mathscr{A}) = \mathbf{SL}(\mathscr{T})$ gilt. □

**Verzweigungs-, Vereinigungs- und Rekurrenzvollständigkeit.** Ist $\mathscr{T}$ ein Spursystem, können wir Zustände ausmachen, an denen eine Verzweigung des Verhaltens von $\mathscr{T}$ in verschiedene Alternativen stattfindet. Einen Zustand, bei dem eine Verzweigung stattfindet, nennen wir einen *Verzweigungspunkt*. Umgekehrt gibt es solche Zustände, die gemeinsames Resultat des Stattfindens verschiedener Verhaltensalternativen sind, die somit identisch fortgesetzt werden können; solche Zustände heißen *Vereinigungspunkte*. Schließlich gibt es spezielle Vereinigungspunkte, die aufgrund zyklischen Verhaltens in einem Spursystem entstehen; solche Vereinigungspunkte nennen wir *Rekurrenzpunkte*.

Wir beginnen unsere Überlegungen mit einem prominenten Beispiel.

III.1.8. BEISPIEL (Speisende Philosophen). $N$ Philosophen ($N > 1$) sitzen zu Tisch. Der Tisch ist kreisförmig und zwischen je zwei Philosophen liegt ein Eßstäbchen (die Verwendung fernöstlichen Tischbestecks hilft uns aus der Schwierigkeit heraus, erklären zu müssen, warum ein vollständiges Eßbesteck aus zwei Gabeln oder Löffeln besteht und warum es unmöglich ist, mit nur einer Gabel oder einem Löffel zu essen). Jeder Philosoph denkt eine Weile nach, nimmt dann zunächst das Stäbchen links und danach das Stäbchen rechts von ihm, um zu speisen. Nachdem sein Appetit einstweilen gestillt ist, legt er beide Stäbchen — zunächst das rechte, danach das linke — an ihren Platz zurück, um wieder in Gedanken zu versinken, bis sein Hunger erneut erwacht. Werden alle Philosophen satt?

Verhalten sich alle Philosophen in der beschriebenen Weise, ist die Frage mit „Nein“ zu beantworten, denn nimmt jeder Philosoph zunächst das auf dem Platz links von ihm liegende Stäbchen, bevor irgendeiner der Denker zu dem Stäbchen rechts von ihm greifen kann, wartet jeder Philsoph darauf, daß sein rechter Nachbar sein linkes Stäbchen auf den Tisch zurücklegt: Die Tischrunde ist in einer Verklemmung gefangen.

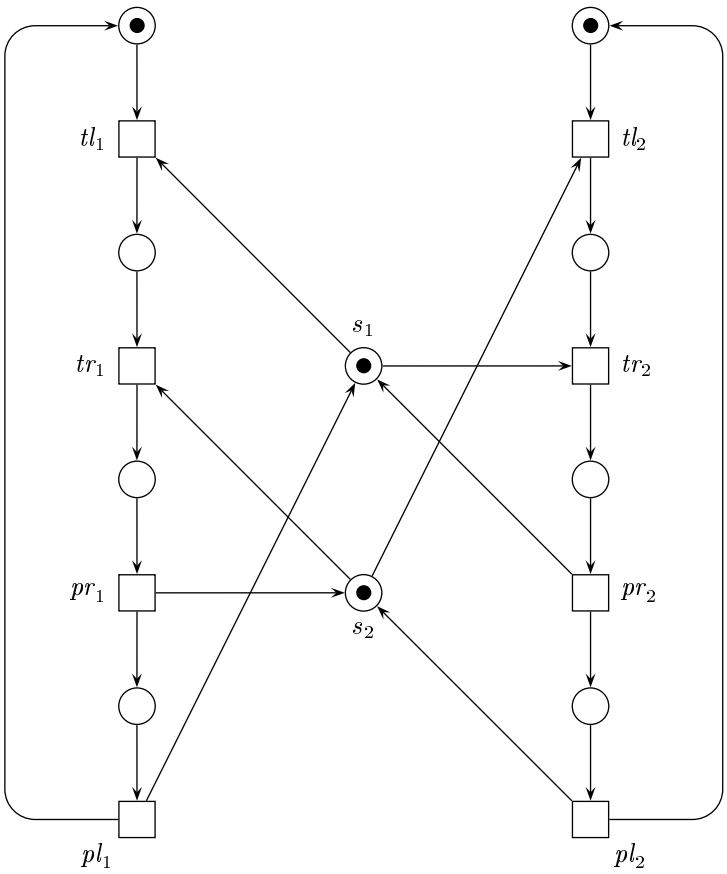



ABBILDUNG III.2. Petri-Netz $\mathcal{N}$ zum Philosophenbeispiel

Ein Petri-Netz, das die Philosophenrunde für $N = 2$ modelliert, ist in Abbildung III.2 dargestellt. Die Transitionen sind für $i = 1, 2$ wie folgt beschriftet: $tl_i$ bzw. $tr_i$ modellieren das Aufnehmen des linken bzw. rechten Stäbchens und $pl_i$ bzw. $pr_i$ das Ablegen des linken bzw. rechten Stäbchens für den $i$-ten Philosophen. Die Plätze $s_1$ und $s_2$ stehen für die beiden verfügbaren Stäbchen, wobei $s_i$ für $i = 1, 2$ markiert ist, wenn sich das $i$-te Stäbchen auf dem Tisch befindet. □

Betrachten wir verschiedene Prozeßautomaten für das Netzsystem $\mathcal{N}$ aus Abbildung III.2. Ein möglicher Prozeßautomat $\mathcal{A}_0$ kann offenbar sofort aus dem Spursystem $\mathcal{T}(\mathcal{N})$ konstruiert werden, indem wir $S_{\mathcal{A}_0} =_{\text{Def}} S_{\mathcal{T}(\mathcal{N})}$, $s_{\mathcal{A}_0} =_{\text{Def}} s_{\mathcal{T}(\mathcal{N})}$, $X_{\mathcal{A}_0} =_{\text{Def}} \{a \in \mathbf{CSO}(\Sigma) : a \in A_\Sigma\}$ und

$$\delta_{\mathcal{A}_0}(s, a) =_{\text{Def}} \delta_{\mathcal{T}(\mathcal{N})}(s, a)$$

wählen (das $a$ auf der linken Seite dieser Festlegung bezeichnet den zu der auf der rechten Seite vorkommenden Aktion $a \in A_\Sigma$ gehörenden Buchstaben $a \in$

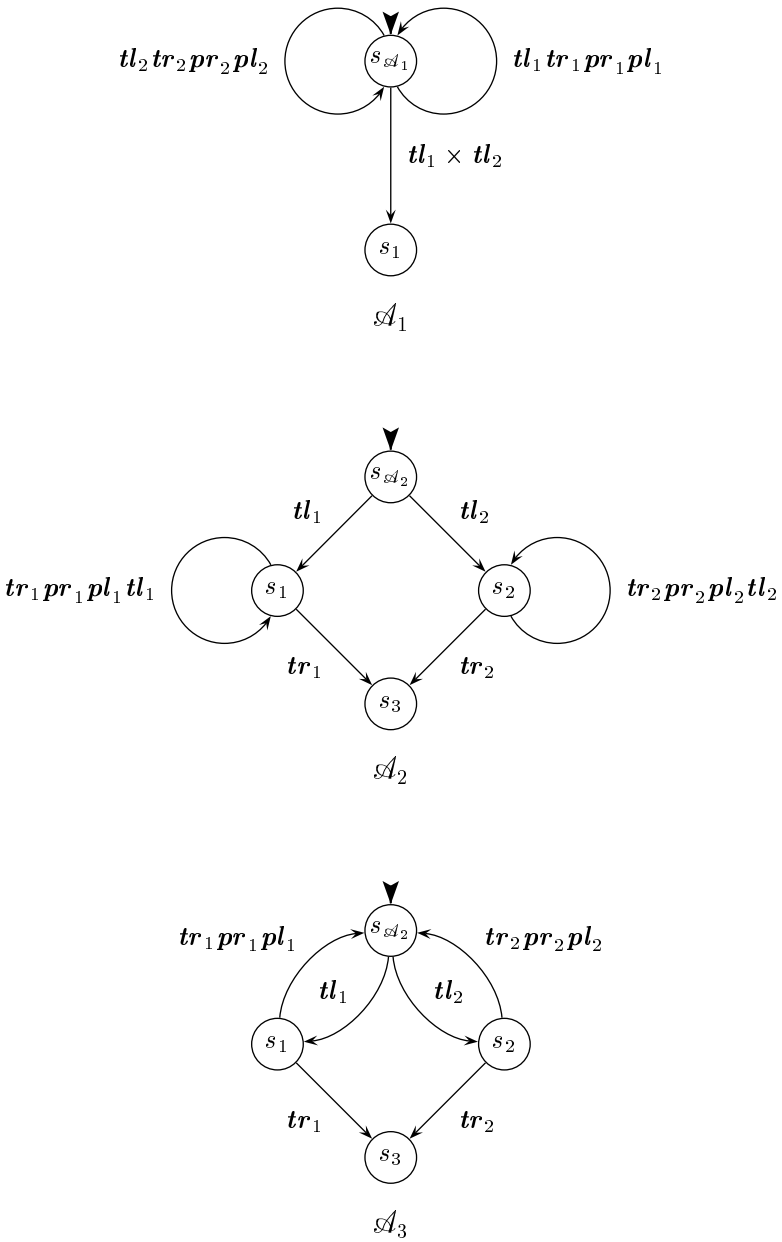



ABBILDUNG III.3. Prozeßautomaten zum Netzsystem in Abbildung III.2.

**CSO**($\Sigma$)). Dieser Prozeßautomat beinhaltet offenbar jeden Verzweigungs-, Vereinigungs- und Rekurrenzpunkt (unabhängig von deren Definition). Allerdings sind auch solche Zustände vorhanden, die nicht in eine der drei Kategorien fallen.

Betrachten wir nun die Prozeßautomaten $\mathcal{A}_1$, $\mathcal{A}_2$ und $\mathcal{A}_3$ in Abb. III.3. Alle drei Automaten sind vollständig bezüglich $\mathcal{T}(\mathcal{N})$. In $\mathcal{A}_1$ werden alle möglichen maximalen azyklischen Verhaltensweisen von $\mathcal{N}$ durch jeweils eine eigene Transition dargestellt: Die beiden Schleifen bezeichnen diejenigen Verhaltensweisen, in denen einer der beiden Philosophen beide Stäbchen ergreift und wieder ablegt, die mit $tl_1 \times tl_2$ bezeichnete Kante repräsentiert die Verhaltensweise, die zu einer Verklemmung führt. Für die Verzweigungen, die nach dem Schalten der Transition $tl_1$ bei der Initialmarkierung von $\mathcal{N}$ möglich ist, existiert kein Zustand in $\mathcal{A}_1$, ebensowenig ist ein Verzweigungspunkt für die beiden möglichen Alternativen nach dem Feuern von $tl_2$ zu finden.

Im Gegensatz dazu enthält $\mathcal{A}_2$ diese beiden Verzweigungspunkte ($s_1$ und $s_2$). Allerdings ist der Rekurrenzpunkt $s_{\mathcal{T}(\mathcal{N})}$ zwar wegen Def. III.1.4.(a) in $\mathcal{A}_2$ enthalten, aber nicht als Vereinigungspunkt der beiden Verhaltensweisen erkennbar, die nicht zu der Verklemmung führen.

$\mathcal{A}_3$ hingegen enthält für jede mögliche Verzweigung im Systemverhalten einen Zustand, bei dem diese Verzweigung an der Existenz verschiedener auslaufender Kanten erkennbar wird. Ebenso ist die Vereinigung im Verhalten des Netzes durch einen entsprechenden Zustand gekennzeichnet.

Wir nennen einen Prozeßautomaten zu einem Spursystem *verzweigungs-*, *vereinigungs-* und *rekurrenzvollständig*, wenn er einen Verzweigungs-, Vereinigungs- und Rekurrenzpunkt für jede mögliche Verzweigung, Vereinigung und Rekurrenz dieses Spursystems enthält und weiterhin die Verzweigung, Vereinigung oder Rekurrenz bei diesem Zustand durch die Existenz verschiedener aus diesem Zustand auslaufende bzw. in diesen Zustand einlaufende Kanten erkennbar ist. Wir nennen einen Prozeßautomaten *konfliktvollständig*, wenn er verzweigungs-, vereinigungs- sowie rekurrenzvollständig ist.

Diese Begriffsbildung ist nicht allein dadurch motiviert, daß die Konfliktstruktur eines Spursystems in einem in diesem Sinne vollständigen Prozeßautomaten offen zutage liegt (tatsächlich werden diese Vollständigkeitseigenschaften in Kapitel IV zur Entwicklung eines Modelcheckers für die temporale Logik DCTL nicht benötigt), sondern im Wesentlichen durch Effizienzüberlegungen. Die Größe eines Prozeßautomaten $\mathcal{A}$ kann in zwei Dimensionen gemessen werden: Die Anzahl seiner Zustände und die Anzahl der Ereignisse, die in Semiordnungen vorkommen, für die Transitionen in $\mathcal{A}$ definiert sind. Ein Prozeßautomat $\mathcal{A}$ etwa, der zwar vollständig bzgl. des zugehörigen Spursystems $\mathcal{T}$ ist, jedoch für einen der Verzweigungspunkte von $\mathcal{T}$ keinen entsprechenden Zustand enthält, muß dennoch eine Kantenverzweigung bei einem Zustand $s$ für einen auftretenden Konflikt zwischen Aktionen $a$ und $b$ enthalten, allerdings müssen $a$ und $b$ nicht simultan bei $s$ schaltfähig sein. D. h., daß dasselbe Verhalten als Präfix verschiedener Semiordnungen auftritt, für

die Transitionen bei $s$ definiert sind; $\mathscr{A}$ enthält mehr als die unbedingt notwendige Anzahl von Ereignissen. Der Zustand $s_{\mathscr{A}_1}$ des Prozeßautomaten $\mathscr{A}_1$ aus Abb. III.2 ist ein Beispiel für einen solchen Zustand. Analoge Überlegungen lassen sich für Vereinigungspunkte bzw. Rekurrenzpunkte anstellen: Enthält ein Prozeßautomat nicht alle Vereinigungspunkte, so wird die Zusammenführung von Verhaltensalternativen „zu lange verzögert". Die Zustände $s_1$ und $s_2$ in $\mathscr{A}_2$ dienen als Beispiele.

Eine Definition der Begriffe Verzweigungs-, Vereinigungs- und Rekurrenzvollständigkeit muß berücksichtigen, daß es Systemaktionen geben kann, die nebenläufig zu der Auswahl einer Verhaltensmöglichkeit bzw. zur Vereinigung verschiedener Alternativen stattfinden. Etwa würde zur Herstellung der $\langle a, b\rangle$-Verzweigungsvollständigkeit das Hinzufügen aller Zustände, bei denen die voneinander abhängigen Aktionen $a$ und $b$ simultan schaltfähig sind, zu einem unnötig großen Prozeßautomaten führen. Wir definieren deshalb zunächst für eine Aktionsmenge $B$ eine Äquivalenzrelation $\simeq_B$ auf den Zuständen eines Spursystems, die solche Zustände in Beziehung setzt, die sich nur durch das Weiterschalten von Aktionen unterscheiden, die nicht in $B$ enthalten sind.

III.1.9. Definition. Sei $\mathcal{T}$ ein Spursystem über $\Sigma$ und seien $B \subseteq A_\Sigma$. Die Relation $\sim_B \subseteq S_\mathcal{T} \times S_\mathcal{T}$ ist als

$$s_1 \sim_B s_2 \Leftrightarrow_{\text{Def}} \exists a \in A_\Sigma \left(\forall b \in B \left(a\ I_\Sigma\ b\ \&\ s_1 \xrightarrow[\mathcal{T}]{a} s_2\right)\right)$$

definiert. $\simeq_B$ bezeichnet die reflexiv-transitive Hülle über dem symmetrischen Abschluß von $\sim_B$, d. h.

$$\simeq_B =_{\text{Def}} \left(\sim_B \cup \sim_B^{-1}\right)^* .$$

Da $\simeq_B$ nun offenbar eine Äquivalenzrelation ist, bezeichnen wir die Äquivalenzklasse eines Zustands $s \in S_\mathcal{T}$ bzgl. $\simeq_B$ mit

$$[s]_B =_{\text{Def}} \{s' \in S_\mathcal{T} : s \simeq_B s'\} .$$

Ist $B = \{a, b\}$, so setzen wir $\simeq_{a,b} =_{\text{Def}} \simeq_B$ und $[s]_{a,b} =_{\text{Def}} [s]_B$. □

Wir kommen nun zur Definition von Verzweigungs-, Vereinigungs- und Rekurrenzpunkten.

III.1.10. Definition (Verzweigungs-, Vereinigungs- und Rekurrenzpunkte). Sei $\mathcal{T}$ ein Spursystem über $\Sigma$ und seien $a, b \in A_\Sigma$ mit $a \neq b$ und $a\ D_\Sigma\ b$. Ein Zustand $s \in S_\mathcal{T}$ heißt $\langle a, b\rangle$-*Verzweigungspunkt* in $\mathcal{T}$, wenn $s \xrightarrow[\mathcal{T}]{a}$ und $s \xrightarrow[\mathcal{T}]{b}$ gilt. $s$ heißt $\langle a, b\rangle$-*Vereinigungspunkt* in $\mathcal{T}$, wenn es Zustände $s_1, s_2 \in S_\mathcal{T}$ mit $s_1 \xrightarrow[\mathcal{T}]{a} s$ und $s_2 \xrightarrow[\mathcal{T}]{b} s$ gibt. Ist $s$ ein $\langle a, b\rangle$-Vereinigungspunkt und gibt es weiterhin Semiordnungen $x, y \in \mathbf{CSO}(\Sigma)$, $x \neq \epsilon$ sowie $y \neq \epsilon$ mit $s_\mathcal{T} \xRightarrow[\mathcal{T}]{x} s \xRightarrow[\mathcal{T}]{y} s$ und

$$\begin{aligned} & (a \in \lambda_x(\max(x))\ \&\ b \in \lambda_y(\max(y))) \\ \vee\ & (b \in \lambda_x(\max(x))\ \&\ a \in \lambda_y(\max(y)))\,, \end{aligned}$$

so nennen wir $s$ einen $\langle a,b\rangle$-*Rekurrenzpunkt* in $\mathcal{T}$.

Wir bezeichnen mit $\mathrm{bp}_{a,b}(\mathcal{T})$ ($\mathrm{jp}_{a,b}(\mathcal{T})$, $\mathrm{rp}_{a,b}(\mathcal{T})$) die Menge der $\langle a,b\rangle$-Verzweigungspunkte ($\langle a,b\rangle$-Vereinigungspunkte, $\langle a,b\rangle$-Rekurrenzpunkte) in $\mathcal{T}$. □

III.1.11. Lemma. *Sei $s \in S_{\mathcal{T}}$ ein Zustand eines Spursystems $\mathcal{T}$ über $\Sigma$.*

(a) $s \in \mathrm{bp}_{a,b}(\mathcal{T}) \Rightarrow [s]_{a,b} \subseteq \mathrm{bp}_{a,b}(\mathcal{T})$,
(b) $s \in \mathrm{jp}_{a,b}(\mathcal{T}) \Rightarrow [s]_{a,b} \subseteq \mathrm{jp}_{a,b}(\mathcal{T})$,
(c) $s \in \mathrm{rp}_{a,b}(\mathcal{T}) \Rightarrow [s]_{a,b} \subseteq \mathrm{rp}_{a,b}(\mathcal{T})$.

III.1.12. Definition (Verzweigungs-, Vereinigungs- und Rekurrenzvollständigkeit). Sei $\mathcal{A}$ ein vollständiger Prozeßautomat zu einem Spursystem $\mathcal{T}$ über dem Spuralphabet $\Sigma$ und seien $a,b \in A_\Sigma$ mit $a \neq b$ und $a\ D_\Sigma\ b$.

(a) $\mathcal{A}$ heißt $\langle a,b\rangle$-*verzweigungsvollständig*, wenn es für alle $s \in \mathrm{bp}_{a,b}(\mathcal{T})$ ein $s' \in S_{\mathcal{A}}$ mit $s \simeq_{a,b} s'$ gibt, so daß Semiordnungen $x,y \in X_{\mathcal{A}}$ und Zustände $s_1, s_2 \in S_{\mathcal{A}}$ mit

$$s' \xrightarrow[\mathcal{A}]{x} s_1,\ s' \xrightarrow[\mathcal{A}]{y} s_2, a \in \lambda_x(\min(x)) \text{ und } b \in \lambda_y(\min(y))$$

existieren.

(b) Wir nennen $\mathcal{A}$ $\langle a,b\rangle$-*vereinigungsvollständig*, wenn es für alle $s \in \mathrm{jp}_{a,b}(\mathcal{T})$ ein $s' \in S_{\mathcal{A}}$ mit $s \simeq_{a,b} s'$ gibt, so daß Semiordnungen $x,y \in X_{\mathcal{A}}$ und Zustände $s_1, s_2 \in S_{\mathcal{A}}$ mit

$$s_1 \xrightarrow[\mathcal{A}]{x} s',\ s_2 \xrightarrow[\mathcal{A}]{y} s', a \in \lambda_x(\max(x)) \text{ und } b \in \lambda_y(\max(y))$$

existieren.

(c) Schließlich ist $\mathcal{A}$ $\langle a,b\rangle$-*rekurrenzvollständig*, wenn es für alle $s \in \mathrm{rp}_{a,b}(\mathcal{T})$ ein $s' \in S_{\mathcal{A}}$ mit $s \simeq_{a,b} s'$ gibt, so daß Semiordnungen $x,y \in X_{\mathcal{A}}$ und Zustände $s_1, s_2 \in S_{\mathcal{A}}$ mit

$$s_1 \xrightarrow[\mathcal{A}]{x} s',\ s_2 \xrightarrow[\mathcal{A}]{y} s', a \in \lambda_x(\max(x)), b \in \lambda_y(\max(y)) \text{ und}$$
$$\left(s' \xRightarrow[\mathcal{A}]{x} s' \vee s' \xRightarrow[\mathcal{A}]{y} s'\right)$$

existieren.

Ist $\mathcal{A}$ für alle $a,b \in A_\Sigma$ mit $a \neq b$ und $a\ D_\Sigma\ b$ $\langle a,b\rangle$-verzweigungsvollständig ($\langle a,b\rangle$-vereinigungsvollständig, $\langle a,b\rangle$-rekurrenzvollständig), so nennen wir $\mathcal{A}$ *verzweigungsvollständig* (*vereinigungsvollständig*, *rekurrenzvollständig*). Ist $\mathcal{A}$ schließlich verzweigungs-, vereinigungs- und rekurrenzvollständig, so heißt $\mathcal{A}$ *konfliktvollständig*. □

III.1.13. Definition (Minimaler Prozeßautomat). Ist $\mathcal{A}$ ein Prozeßautomat zu einem Spursystem $\mathcal{T}$, so nennen wir $\mathcal{A}$ *minimal*, wenn $\mathcal{A}$ konfliktvollständig ist und eine minimale Zahl von Zuständen unter allen konfliktvollständigen Prozeßautomaten zu $\mathcal{T}$ aufweist. □

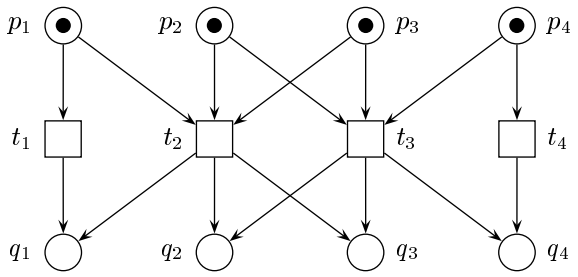



ABBILDUNG III.4. Ein globaler Vereinigungspunkt.

III.1.14. BEMERKUNG (Nicht lokale Vereinigunspunkte). Leser, die mit der Theorie der *verzweigten Abläufe* und ihrer endlichen Repräsentation, der Auffaltung vertraut sind, mögen sich die Frage nach dem Verhältnis von Vereinigungspunkten und *cut-off*-Ereignissen stellen. Wir können eine Semiordnung $x$ als *lokale Konfiguration* ansehen, wenn $|\max(x)| \leqslant 1$ ist ($\epsilon$ ist eine lokale Konfiguration). Zur Definition von *cut-off*-Ereignissen sind verschiedene Eigenschaften (vgl. [**31, 53**]) notwendig, von denen an dieser Stelle jedoch nur die folgende interessant ist: Ist $e \in \max(x)$ ein *cut-off*-Ereignis, dann gibt es eine weitere lokale Konfiguration $y$ mit $x \not\equiv y$, so daß $\delta_{\mathcal{T}}(s_{\mathcal{T}}, x) = \delta_{\mathcal{T}}(s_{\mathcal{T}}, y)$ gilt. Ist $e' \in \max(y)$, so ist $\delta_{\mathcal{T}}(s_{\mathcal{T}}, x) \in \mathrm{jp}_{\lambda_x(e),\lambda_y(e')}(\mathcal{T})$. Allerdings gibt es Vereinigungspunkte, die nicht lokal, d. h. nicht durch das ausschließliche Stattfinden von Ereignissen in der Vergangenheit von $e$ bzw. $e'$ erreichbar sind. Ein Beispiel ist in Abb. III.4 gezeigt. Keine der vier Transitionen bestimmt ein *cut-off*-Ereignis, der Zustand $\{q_i : 1 \leqslant i \leqslant 4\}$ ist jedoch ein $t_2, t_3$-Vereinigungspunkt. □

## III.2. Erzeugung von Prozeßautomaten

In diesem Abschnitt diskutieren wir einen Algorithmus zur Konstruktion eines Prozeßautomaten $\mathcal{A}$ zu einem Spursystem $\mathcal{T}$ über einem Spuralphabet $\Sigma$. Wir zeigen, daß $\mathcal{A}$ vollständig bzgl. $\mathcal{T}$ ist.

Der als Algorithmus III.1 abgebildete funktioniert wie folgt: Es finden zwei Mengen $S_{\mathcal{A}}$ und $Q$ als Datenstrukturen Verwendung. $S_{\mathcal{A}}$ enthält Zustände von $\mathcal{T}$, die bei der Erzeugung von $\mathcal{A}$ generiert werden; diese Menge wird nach Termination des Algorithmus die Zustandsmenge von $\mathcal{A}$ bilden. $Q$ enthält Zustände von $\mathcal{A}$, die noch nicht vollständig behandelt wurden.

Ein Zustand $s$ gilt als behandelt, wenn alle Transitionsübergänge der Form $\delta_{\mathcal{A}}(s, x)$ erzeugt wurden, die der Prozeßautomat nach Termination des Algorithmus enthält. Initial enthalten $Q$ und $S_{\mathcal{A}}$ lediglich den Anfangszustand von $\mathcal{T}$. Solange $Q$ noch unbehandelte Zustände enthält, wird einer dieser Zustände $s$ ausgewählt. Es wird eine Menge von Semiordnungen $x_1, x_2, \ldots, x_k$ erzeugt, die bei diesem Zustand konzessioniert sind, sowie die Folgezustände $s_i$ $(1 \leqslant i \leqslant k)$, die sich aus dem

**algorithm** *generate* **is**
  **input** $\mathcal{T}$, ein Spursystem über $\Sigma$;
  **output** $\mathcal{A}$, ein Prozeßautomat zu $\mathcal{T}$;
  **local variables** $Q \in \mathcal{P}(S_{\mathcal{T}})$; $s, s' \in S_{\mathcal{T}}$;
    $C \in \mathcal{P}(A_\Sigma)$; $x \in \mathbf{CSO}(\Sigma)$;
**begin**
(1) $s_{\mathcal{A}} \leftarrow s_{\mathcal{T}}$; $S_{\mathcal{A}} \leftarrow \{s_{\mathcal{A}}\}$; $X_{\mathcal{A}} \leftarrow \varnothing$; $\delta_{\mathcal{A}} \leftarrow \varnothing$; $Q \leftarrow \{s_{\mathcal{A}}\}$;
(2) **while** $Q \neq \varnothing$ **do**
(3)  select $s \in Q$; $Q \leftarrow Q - \{s\}$;
(4)  **foreach** $C \in \mathit{steps}(s)$ **do**
(5)  $x \leftarrow \mathit{so}(C)$; $s' \leftarrow \delta_{\mathcal{T}}(s, x)$; $\mathit{extend}(x, s')$;
(6)  **if** $s' \notin S_{\mathcal{A}}$ **then** $Q \leftarrow Q \cup \{s'\}$; $S_{\mathcal{A}} \leftarrow S_{\mathcal{A}} \cup \{s'\}$ **fi**;
(7)  $X_{\mathcal{A}} \leftarrow X_{\mathcal{A}} \cup \{x\}$; $\delta_{\mathcal{A}}(s, x) \leftarrow s'$
(8)  **od**
(9) **od**
**end** *generate*;

ALGORITHMUS III.1. Grundalgorithmus zur Konstruktion eines Prozeßautomaten.

Schalten von $x_i$ bei $s$ ergeben. Diese Semiordnungen und ihre Folgezustände werden als Zustandsübergänge $s \xrightarrow[\mathcal{A}]{x_i} s_i$ zu $\mathcal{A}$ hinzugefügt, ist $s_i$ überdies noch nicht in $S_{\mathcal{A}}$ vorhanden, so ist $s_i$ noch nicht behandelt worden und wird deshalb in $S_{\mathcal{A}}$ und $Q$ eingefügt. $s$ ist nun behandelt und kann aus $Q$ entfernt werden.

Wir haben die folgenden Probleme zu behandeln:

(P1) Ist $s$ ein bereits erzeugter Zustand von $\mathcal{A}$, wie konstruieren wir eine adäquate Menge von Semiordnungen $x_1, x_2, \ldots, x_k$, für die Folgezustände $\delta_{\mathcal{A}}(s, x_i)$ bestimmt werden können?

(P2) Unter welchen Bedingungen können weitere Ereignisse zu einer Semiordnung $x$, die bei $s$ konzessioniert ist, hinzugefügt werden. Wann terminiert die Konstruktion einer Semiordnung?

**Zu Problem (P1): Teil I.** Ist $s$ ein bereits konstruierter Zustand von $\mathcal{A}$, müssen wir für die bei $s$ stattfindenden Transitionen Semiordnungen konstruieren. Dabei ist offenbar sicherzustellen, daß jede bei $s$ konzessionierte Aktion als Ereignisbeschriftung in einer dieser Semiordnungen vorkommt. Problematischer jedoch ist, daß bei $s$ *alternative Verhaltensweisen* konzessioniert sein können, d. h. es gibt Semiordnungen $x$ und $y$, so daß $s \overset{x}{\Longrightarrow}_{\mathcal{T}}$ und $s \overset{y}{\Longrightarrow}_{\mathcal{T}}$ gilt, jedoch nicht $s \overset{z}{\Longrightarrow}_{\mathcal{T}}$ für irgendein $z \in \boldsymbol{x} \vee \boldsymbol{y}$; an diesem Punkt müssen dann unterschiedliche Transitionsübergänge in $\mathcal{A}$ erzeugt werden (vgl. die Diskussion von Problem (P2)). Diese „globale“ Charakterisierung alternativen Verhaltens ist allerdings nicht hilfreich. Wir benötigen

an dieser Stelle eine Charakterisierung, die auf dem Verhältnis einzelner Aktionen beruht.

Ein erster Ansatz wäre: Es gibt alternative Verhaltensweisen bei $s$, wenn es bei $s$ konzessionierte Aktionen $a$ und $b$ mit $a \neq b$ und $a\ D_\Sigma\ b$ gibt; formal:

III.2.1. DEFINITION. Sei $\mathcal{T}$ ein Spursystem über $\Sigma$ und sei $s \in S_\mathcal{T}$. Die Relation $D^{\rightarrow}_\Sigma(s) \subseteq D_\Sigma$ ist als

$$a\ D^{\rightarrow}_\Sigma(s)\ b \Leftrightarrow_{\mathrm{Def}} s \in \mathrm{bp}_{a,b}(\mathcal{T})$$

definiert. □

Man beachte, daß $D^{\rightarrow}_\Sigma(s)$ im Gegensatz zu $D_\Sigma$ irreflexiv ist. Weiterhin gilt offenbar $D^{\rightarrow}_\Sigma(s) = (D_\Sigma \cap (\mathrm{en}_\mathcal{T}(s) \times \mathrm{en}_\mathcal{T}(s))) - \mathrm{id}_{A_\Sigma}$.

Betrachten wir nun die folgende Strategie zur Erzeugung von Semiordnungen $x$, die als Transitionsübergänge bei $s$ in $\mathcal{A}$ auftreten: Zur Bestimmung der Menge der minimalen Elemente solcher Semiordnungen $x$ erzeugen wir die Menge aller maximalen Schritte in $\mathrm{en}_\mathcal{T}(s)$, d. h. die Menge der maximalen Cliquen in $\mathrm{en}_\mathcal{T}(s)$ bzgl. der Relation

$$I =_{\mathrm{Def}} (A_\Sigma \times A_\Sigma) - D^{\rightarrow}_\Sigma(s),$$

dem Komplement von $D^{\rightarrow}_\Sigma(s)$. Ist $C$ eine solche Clique, konstruieren wir nun eine Semiordnung $x \equiv \langle C, \varnothing, \mathrm{id}_C \rangle$. Anschließend fügen wir unter Verwendung der Operation $\circ_\Sigma$ Ereignisse für noch zu bestimmende unter $\delta_\mathcal{A}(s,x)$ konzessionierte Aktionen zu $x$ hinzu, bis eine noch zu diskutierende Terminationsbedingung eintritt (Problem (P2)).

Diese Strategie ist jedoch aufgrund der Existenz sog. Konfusion ungeeignet für die Konstruktion eines vollständigen Prozeßautomaten für $\mathcal{T}$. Unter einer *Konfusion* verstehen wir die folgende Situation: Sei $s \in S_\mathcal{T}$ ein Zustand eines Spursystems $\mathcal{T}$ über $\Sigma$. Bei $s$ liegt eine Konfusion vor, wenn es $a, b, c \in A_\Sigma$ mit $a\ I_\Sigma\ b$ gibt, so daß $s \xrightarrow[\mathcal{T}]{ac}$, $s \xrightarrow[\mathcal{T}]{a \times b}$, jedoch $\neg s \xrightarrow[\mathcal{T}]{(a \times b)c}$ gilt.

III.2.2. BEISPIEL. Betrachten wir das Netzsystem $\mathcal{N}$ in Abb. III.5 und die beiden Prozeßautomaten $\mathcal{A}_1$ und $\mathcal{A}_2$: Bei der Markierung $Q_\mathcal{N}$ existiert der maximale Schritt $C = \{a, b\}$. Nach der oben beschriebenen Strategie würde der Prozeßautomat $\mathcal{A}_1$ konstruiert. $\mathcal{A}_1$ ist jedoch im Gegensatz zu $\mathcal{A}_2$ unvollständig bzgl. $\mathcal{T}(\mathcal{N})$.[2] □

Wir benötigen also eine größere Relation $F$ als $D^{\rightarrow}_\Sigma(s)$ zur Erzeugung maximaler Schritte bei $s$. Zunächst stellen wir fest, daß eine solche Relation $F$ unabhängig

[2] Unser Grundalgorithmus erzeugt allerdings (unter Verwendung der Vorwärtskonfliktrelation $F = D^{\rightarrow}_\Sigma$, s.u.) nicht den Prozeßautomaten $\mathcal{A}_2$, sondern einen redundanten Prozeßautomaten, der neben den Zuständen und Zustandsübergängen von $\mathcal{A}_2$ auch den Übergang für $a \times b$ enthält. Die in Abschnitt III.3 diskutierte Reformulierung des Grundalgorithmus erzeugt jedoch für dieses Beispiel den Prozeßautomaten $\mathcal{A}_2$.

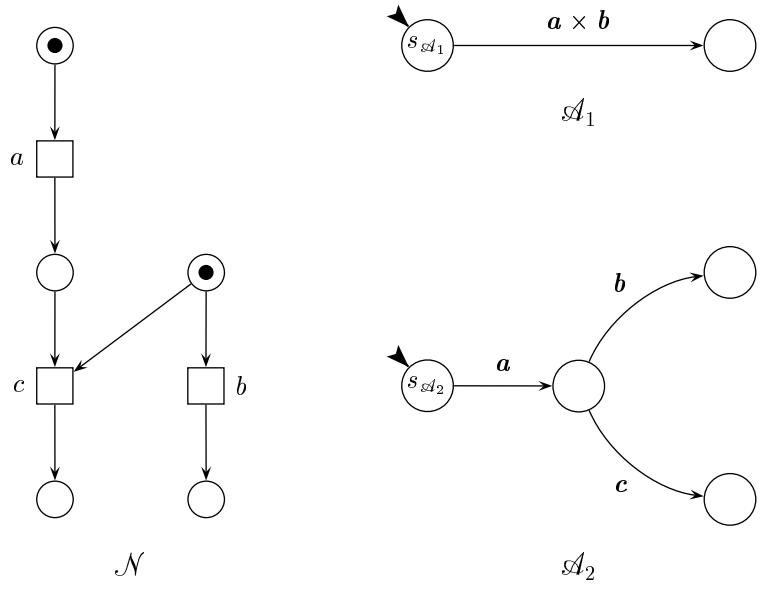



ABBILDUNG III.5. Konfusionssituation.

vom Zustand $s$ sein muß, da eine Konfusionssituation nicht als direkter Konflikt bei $s$ auftritt. Die Definition

$$a\ F\ b \Leftrightarrow_{\text{Def}} \exists s \in S_{\mathcal{T}}\,(a\ D_\Sigma^{\rightarrow}(s)\ b)$$

ist jedoch problematisch, da während der Erzeugung des Prozeßautomaten bereits bekannt sein müßte, ob $a$ und $b$ bei irgendeinem (u. U. noch nicht behandelten) Zustand von $\mathcal{T}$ simultan konzessioniert sind. Dies kann i. Allg. nicht ohne vollständige Erzeugung aller Zustände von $\mathcal{T}$ festgestellt werden. Wir verwenden deshalb die folgende schwächere Definition:

III.2.3. DEFINITION. Sei $\mathcal{T}$ ein Spursystem über $\Sigma$. Die Relation $D_\Sigma^{\rightarrow} \subseteq A_\Sigma \times A_\Sigma$ ist als

$$D_\Sigma^{\rightarrow} = \bigcup_{s \in S_{\mathcal{T}}} D_\Sigma^{\rightarrow}(s)$$

definiert. I. Allg. bezeichnen wir jede symmetrische und irreflexive Relation $F \subseteq A_\Sigma \times A_\Sigma$, die die Bedingung $F \supseteq D_\Sigma^{\rightarrow}$ erfüllt, als *Vorwärtskonfliktrelation* in $\mathcal{T}$. □

Wir verwenden die folgende Strategie: Sei $F$ eine Vorwärtskonfliktrelation in $\mathcal{T}$. Zur Konstruktion von bei einem Zustand $s$ von $\mathcal{T}$ konzessionierten Semiordnungen, der bei der Erzeugung eines Prozeßautomaten $\mathcal{A}$ von $\mathcal{T}$ behandelt wird, verwenden wir die Relation

$$a\ I(s)\ b \Leftrightarrow_{\text{Def}} a\ \bar{F}\ b\ \&\ F(a) \subseteq \text{en}_{\mathcal{T}}(s)\ \&\ F(b) \subseteq \text{en}_{\mathcal{T}}(s).$$

zur Erzeugung maximaler Schritte.

Die in Beispiel III.2.2 dargestelle Konfusionssituation wird nun wie folgt behandelt: Unter der Relation $I(s)$ wird für eine Aktion $b$ ein Einzelschritt erzeugt (d. h. ein Schritt der Form $C = \{b\}$), wenn es eine weitere Aktion $c$ gibt, die sich in Vorwärtskonflikt zu $c$ befindet und die bei $s$ keine Konzession hat.

Nun wandeln wir wie oben beschrieben einen unter $I(s)$ erzeugten maximalen Schritt $C$ in eine Semiordnung $x$ um. Da $x$ — wie wir bei unserer Diskussion von Problem (P2) sehen werden — lediglich um solche Ereignisse $e$ erweitert wird, für deren Beschriftungen $F(\lambda_x(e)) = \varnothing$ gilt, wird $c$ unter der Voraussetzung, daß es einen Zustand $s'$ gibt, bei dem $c$ konzessioniert ist, irgendwann Konzession erhalten.

**Zu Problem (P1): Teil II.** Wir haben in unserer Diskussion von Problem (P1) allerdings folgende Situation übersehen: Sei $s \in S_{\mathcal{F}}$ ein Zustand, der bei der Konstruktion eines Prozeßautomaten $\mathcal{A}$ erzeugt wird. Nehmen wir eine Semiordnung $y \in \mathbf{CSO}(\Sigma)$ und eine Aktion $a \in A_\Sigma$ an, so daß $s \overset{y \circ_\Sigma a}{\underset{\mathcal{F}}{\Longrightarrow}}$ gilt, weiterhin sei $y$ minimal bzgl. $\leqslant$, d. h. es gibt ein eindeutig bestimmtes Ereignis $\{e\} = \max(y \circ_\Sigma a)$ mit $\lambda_y(e) = a$; wir konstruieren einen Fall, in dem $\boldsymbol{y} \in \mathbf{SL}(\mathcal{A})$, jedoch $[y \circ_\Sigma a] \notin \mathbf{SL}(\mathcal{A})$ ist.

Sei $b \in A_\Sigma$ eine weitere Aktion mit $a \; D_\Sigma \; b$, wobei wir — um die Diskussion überschaubar zu halten — $D_\Sigma(a) = \{b\}$ annehmen; der Fall $D_\Sigma(a) \supseteq \{b\}$ kann ganz analog behandelt werden. Untersuchen wir den Schritt $C$ genauer, der bei $s$ erzeugt wird und für den $\lambda_y(\min(y)) \subseteq C$ gilt. Gilt $a \notin \mathrm{en}_{\mathcal{F}}(s)$, so wird für $b$ ein Einzelschritt konstruiert; es gilt entweder $C = \{b\}$ oder $b \notin C$. In beiden Fällen raubt $b$ der Aktion $a$ nach dem Feuern von $y$ nicht die Konzession.

Gilt allerdings $a \in \mathrm{en}_{\mathcal{F}}$, ist es möglich, daß $b \in C - \lambda_y(\min(y))$ und wegen $a \; D_\Sigma \; b$ natürlich $a \notin C$ gilt, d. h. insbesondere ist $|C| > 1$. Zunächst stellen wir fest, daß dann $y = y[\min(y)]$ gilt: Gäbe es nämlich für ein $\lambda_y(e) \in C - \{b\}$ ein Ereignis $e' \in E_y$ mit $e \lessdot_y e'$, so daß $\lambda_y(e')$ bei $s$ nicht schaltfähig wäre, würde wegen $\lambda_y(e) \; D_\Sigma \; \lambda_y(e')$ für $\lambda_y(e)$ bei $s$ ein Einzelschritt erzeugt werden.

Weiterhin ist $a$ bei dem Zustand $\delta_{\mathcal{F}}(s, y[C - \{b\}])$ schaltfähig, bei $\delta_{\mathcal{F}}(s, y[C])$ jedoch nicht. Abb. III.6 stellt ein Spursystem und ein Netzsystem für diese Situation dar: Unter der in Teil I diskutierten Strategie wird der entstehende Prozeßautomat das Semiwort ***ca*** nicht als Element seiner Semisprache enthalten.

Zum Zustandekommen dieser Situation ist es jedoch offenbar notwendig, daß $a$ bei $s$ *und* bei $\delta_{\mathcal{F}}(s, y[C - \{b\}])$ konzessioniert ist. Verwenden wir also eine Relation $\mathrm{per} \subseteq A_\Sigma \times A_\Sigma$ mit

$$c \; \mathrm{per} \; a \Rightarrow c \; F \; a \; \& \; \forall s_1, s_2 \in S_{\mathcal{F}} \left( s_1 \overset{c}{\underset{\mathcal{F}}{\rightarrow}} s_2 \; \& \; a \in \mathrm{en}_{\mathcal{F}}(s_1) \Rightarrow a \in \mathrm{en}_{\mathcal{F}}(s_2) \right)$$

und fordern wir, daß Aktionen $c$ mit $\mathrm{per}(c) \neq \varnothing$ als Einzelschritte behandelt werden; die Redefinition der Relation $I(s)$ ist dann

$$\begin{aligned} a \; I(s) \; b \quad \Leftrightarrow_{\mathrm{Def}} \quad & a \; \bar{F} \; b \; \& \; F(a) \subseteq \mathrm{en}_{\mathcal{F}}(s) \; \& \; F(b) \subseteq \mathrm{en}_{\mathcal{F}}(s) \\ & \& \; \mathrm{per}(a) = \varnothing \; \& \; \mathrm{per}(b) = \varnothing. \end{aligned}$$

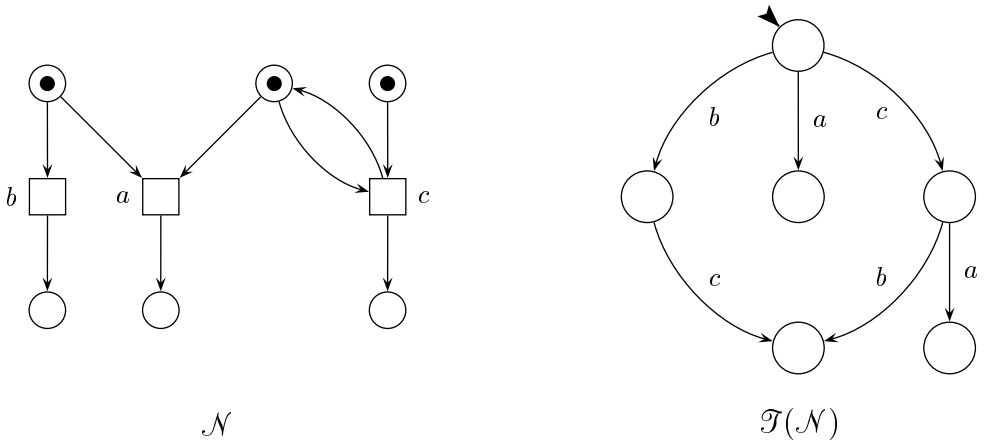



ABBILDUNG III.6. Illustration zum Teil II der Diskussion von Problem (P1).

---

III.2.4. BEMERKUNG. Für viele Netzsysteme ist

$$t_1 \text{ per } t_2 \Leftrightarrow_{\text{Def}} \dot{t}_2 \cap (\dot{t}_1 \cap \dot{t}_2^{\cdot}) \neq \varnothing$$

eine akzeptable Wahl. □

III.2.5. BEMERKUNG. Vernadat et. al. [**93, 94**] verfolgen zur Erzeugung von *step covering graphs* eine ähnliche Strategie, allerdings wird anstelle von $F$ und per eine transitive Vorwärtskonfliktrelation verwendet; dieser Umgang mit der in Teil II unserer Diskussion zu Problem (P2) dargestellten Situation erzeugt aber offenbar kleinere Schritte als unsere Lösung des Problems. □

**Zu Problem (P2).** Wir beweisen zwei Theoreme, die uns zu einer Lösung für dieses Problem führen werden. Vorbereitend benötigen wir den Begriff des *Kreises* in einem Spursystem $\mathcal{T}$.

III.2.6. DEFINITION. Sei $\mathcal{T}$ ein Spursystem über $\Sigma$. Ein *Weg* in $\mathcal{T}$ ist eine Folge $\alpha = s_0, a_0, s_1, a_1, \ldots, s_{n-1}, a_{n-1}, s_n$ für ein $n \geqslant 0$, so daß $a_0, a_1, \ldots, a_{n-1} \in A_\Sigma$ Aktionen und $s_0, s_1, \ldots, s_n \in S_\mathcal{T}$ Zustände mit

$$s_0 \xrightarrow[\mathcal{T}]{a_0} s_1 \xrightarrow[\mathcal{T}]{a_1} \cdots \xrightarrow[\mathcal{T}]{a_{n-1}} s_n$$

sind. $\alpha$ heißt *Kreis* in $\mathcal{T}$, wenn $s_0 = s_n$ gilt, und *Elementarkreis*, wenn $\alpha$ ein Kreis mit der Eigenschaft $i \neq j \Rightarrow s_i \neq s_j$ für $0 < i, j < n$ ist. Ist $s = s_i$ für $0 \leqslant i \leqslant n$, so sagen wir, daß $s$ *auf dem Weg (Kreis, Elementarkreis)* $\alpha$ *liegt* bzw. daß $\alpha$ *über* $s$ *verläuft.* □

---

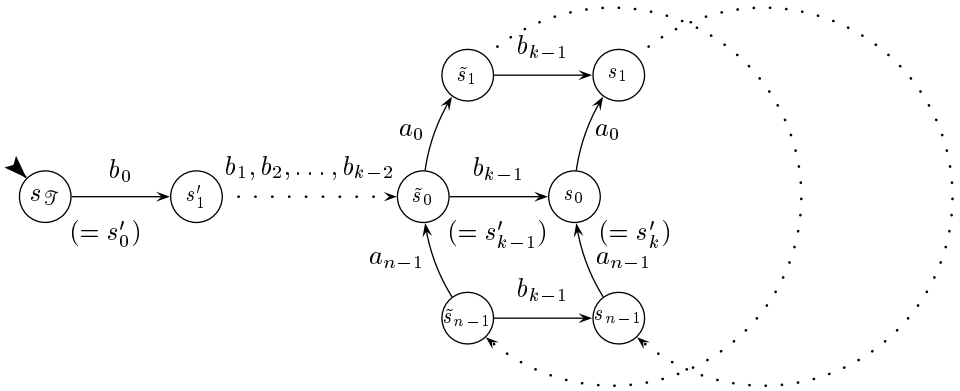



ABBILDUNG III.7. Illustration zum Beweis von Theorem III.2.7.

---

III.2.7. THEOREM. *Sei $\mathcal{J}$ ein wohlgeformtes Spursystem über $\Sigma$. Dann verläuft jeder Elementarkreis $\alpha = s_0, a_0, s_1, a_1, \ldots, s_{n-1}, a_{n-1}, s_n$ über einen Zustand $s_i$ ($0 \leqslant i \leqslant n$), so daß $s_i$ ein $\langle a_{i-_n 1}, b\rangle$-Rekurrenzpunkt für ein $b \in A_\Sigma$ ist. Dabei bezeichnet $i -_n 1$ das Dekrement modulo $n$, d. h.*

$$i -_n 1 =_{Def} \begin{cases} i-1, & \text{falls } i > 0; \\ n-1, & \text{sonst.} \end{cases}$$

BEWEIS. Abbildung III.7 illustriert die folgenden Konstruktion. Wir wählen einen Elementarkreis

$$\alpha = s_0, a_0, s_1, a_1, \ldots, s_{n-1}, a_{n-1}, s_n$$

in $\mathcal{J}$, der nicht über einen $\langle a_{i-_n 1}, b\rangle$-Rekurrenzpunkt wie im Theorem beschrieben verläuft. Sei

$$\beta = s'_0, b_0, s'_1, b_1, \ldots, s'_{k-1}, b_{k-1}, s'_k$$

ein Weg in $\mathcal{J}$, so daß $s'_0 = s_{\mathcal{J}}$ und $s'_k = s_i$ für ein $i$ mit $0 \leqslant i \leqslant n$ gilt. Da $\mathcal{J}$ initialisiert ist, existiert ein solcher Weg mit $k > 0$: Es gilt $b_0 = a_I$, wobei $a_I$ die Initialisierungsaktion von $\mathcal{J}$ ist. Wählen wir $\beta$ und $\alpha$ so, daß $k$ minimal ist, d. h. die Zustände $s'_0, s'_1, \ldots, s'_{k-1}$ liegen nicht auf einem Elementarkreis. Der Einfachheit halber nehmen wir $i = 0$ an; ist dies nicht der Fall, können wir o. B. d. A. die Nummerierung der Elemente von $\alpha$ ändern.

Da $s_0$ auf $\alpha$ liegt, folgt $b_{k-1}\ I_\Sigma\ a_{n-1}$, da ansonsten $s_0 \in \mathrm{rp}_{\langle b_{k-1}, a_{n-1}\rangle}(\mathcal{J})$ wäre.

$k = 1$ stellt sich als unmöglich heraus, da dies $b_{k-1} = a_I$ implizieren würde. Damit gilt jedoch $s_{n-1} \xrightarrow[\mathcal{J}]{a_{n-1}} s_{\mathcal{J}}$ nach Definition I.3.12.(b). Dies widerspricht jedoch Definition I.3.51.(c): $\mathcal{J}$ wäre nicht initialisiert.

---

Sei also $k > 0$. Wir haben bereits festgestellt, daß $b_{k-1}\ I_\Sigma\ a_{n-1}$ gilt; insbesondere gibt es jedoch nach Definition I.3.12.(b) einen Zustand $\tilde{s}_{n-1} \in S_\mathcal{T}$ mit $\tilde{s}_{n-1} \xrightarrow[\mathcal{T}]{a_{n-1}} s'_{k-1}$ und $\tilde{s}_{n-1} \xrightarrow[\mathcal{T}]{b_{k-1}} s_{n-1}$.

Wiederholen wir diese Argumentation für $a_{n-2}$ und $b_{k-1}$, erhalten wir $b_{k-1}\ I_\Sigma\ a_{n-2}$ und können weiterhin die Existenz eines Zustands $\tilde{s}_{n-2}$ mit $\tilde{s}_{n-2} \xrightarrow[\mathcal{T}]{a_{n-2}} \tilde{s}_{n-1}$ und $\tilde{s}_{n-2} \xrightarrow[\mathcal{T}]{b_{k-1}} s_{n-2}$ nachweisen.

Es ist also möglich, für jeden der Zustände $s_{n-i}$ $(0 \leqslant i \leqslant n)$ einen entprechenden Zustand $\tilde{s}_{n-i}$ mit $\tilde{s}_{n-i} \xrightarrow[\mathcal{T}]{b_{k-1}} s_{n-i}$ und $\tilde{s}_{n-i} \xrightarrow[\mathcal{T}]{a_{n-i}} \tilde{s}_{n-i+1}$ zu finden (dabei war $\tilde{s}_0 = s'_{k-1}$). Insbesondere ist $\tilde{s}_0, a_0, \tilde{s}_1, a_1, \ldots, \tilde{s}_{n-1}, a_{n-1}, \tilde{s}_n$ ein Elementarkreis in $\mathcal{T}$, für den es einen Weg $s_\mathcal{T} = s'_0, b_0, s'_1, b_1, \ldots, s'_{k-1} = \tilde{s}_0$ gibt. Die Existenz eines derartigen Kreises und eines derartigen Weges widerspricht jedoch der Minimalität von $k$. □

Theorem III.2.7 liefert uns noch kein brauchbares Terminationskriterium für die Konstruktion von Semiordnungen bei einem während der Prozeßautomatenerzeugung betrachteten Zustand. Ein solches Kriterium läßt sich jedoch aus der folgenden Eigenschaft von Semiwörtern $\boldsymbol{x} \in \mathbf{LSSL}(\mathcal{T})$ herleiten.

III.2.8. Theorem. *Sei $\mathcal{T}$ ein wohlgeformtes Spursystem über $\Sigma$ und sei $a_I \in A_\Sigma$ die Initialisierungsaktion von $\mathcal{T}$. Ist $x_0, x_1, \ldots$ eine unendliche Folge von Semiordnungen mit $\boldsymbol{x}_i \in \mathbf{LSSL}(\mathcal{T})$ und $x_i \overset{a}{\Longrightarrow} x_{i+1}$ für ein $a \in A_\Sigma$ $(0 \leqslant i)$, so existiert ein $k \geqslant 0$, so daß jedes $e \in \max(x_k)$ eine der beiden folgenden Eigenschaften aufweist:*

(a) $<_{x_j}\left(H^{x_j}_{x_k}(e)\right) = \varnothing$ *für alle $j \geqslant k$, oder*
(b) *es gibt ein $b \in A_\Sigma$ und eine unabhängige Menge $C \subseteq E_{x_k}$ mit $e \in C$, so daß $\delta^{x_k}_\mathcal{T}\left(s_\mathcal{T}, \leqslant^{-1}_{x_k}(C)\right)$ ein $\langle b, \lambda_{x_k}(e)\rangle$-Rekurrenzpunkt ist.*

Beweis. Wählen wir ein $n \geqslant 0$ so, daß für alle $e \in \max(x_n)$ gilt:

(i) $<_{x_j}\left(H^{x_j}_{x_n}(e)\right) = \varnothing$ für alle $j \geqslant n$, oder
(ii) es gibt ein $e' \in E_{x_n}$ mit $e' <_{x_n} e$, $\lambda_{x_n}(e') = \lambda_{x_n}(e)$ und $\delta^{x_n}_\mathcal{T}\left(s_\mathcal{T}, \leqslant^{-1}_{x_n}(e')\right) = \delta^{x_n}_\mathcal{T}\left(s_\mathcal{T}, \leqslant^{-1}_{x_n}(e)\right)$; in diesem Fall setzen wir $s =_{\text{Def}} \delta^{x_n}_\mathcal{T}\left(s_\mathcal{T}, \leqslant^{-1}_{x_n}(e)\right)$

Da $\mathcal{T}$ endlich ist, läßt sich immer ein solches $x_n$ in der unendlichen Folge $x_0, x_1, \ldots$ finden.

Wir stellen nun eine Menge $M \subseteq E_{x_n}$ zusammen, die die maximalen Ereignisse in einer Semiordnung $x \equiv x_k$ enthalten wird. Zunächst sind alle Ereignisse $e$ in $M$ enthalten, die die Bedingung (i) erfüllen.

Gilt für $e$ und $e'$ die Bedingung (ii), so betrachten wir nun die Semiordnung $y =_{\text{Def}} x_n\left[\leqslant^{-1}_{x_n}(e) - \leqslant^{-1}_{x_n}(e')\right]$. Es gilt offenbar $s \overset{y}{\Longrightarrow}_\mathcal{T} s$; weiterhin definiert jede Linearisierung von $y$ einen Kreis $\alpha$ in $\mathcal{T}$. Da jeder Kreis $\alpha$ von $\mathcal{T}$ als eine Folge von Elementarkreisen dargestellt werden kann, enthält $\alpha$ nach Theorem III.2.7

wenigstens einen Rekurrenzpunkt $s'$. Sei $\sigma$ eine Schaltfolge mit $s \stackrel{\sigma}{\underset{\mathcal{T}}{\Longrightarrow}} s'$, so daß $\sigma$ die Linearisierung eines Präfix $z$ von $y$ ist. Dann ist $H_z^y(\max(z)) \subseteq M$; $H_z^y(\max(z))$ entspricht der unabhängigen Menge $C$ im Theorem. Damit sind alle Elemente von $M$ bestimmt.

Setzen wir $x =_{\text{Def}} x_n \left[\leqslant_{x_n}^{-1}(M)\right]$. Offensichtlich erfüllt $x$ die Anforderungen an $x_k$. Um nachzuweisen, daß es ein $k$ mit $x \equiv x_k$ gibt, machen wir uns klar, daß es unabhängig von der Auswahl der Aktionen $a_0, a_1, \ldots$ ein $l \geqslant 0$ gibt, so daß $x_k = a_0 \circ_\Sigma a_1 \circ_\Sigma \ldots \circ_\Sigma a_l$ und $x_l \equiv x$ ist. Wir setzen $k = l$. □

Inwieweit ist Theorem III.2.8 nun bei der Suche nach einem Terminationskriterium für die Erzeugung von Semiordnungen bei der Prozeßautomatengenerierung hilfreich? Die Konstruktion einer Semiordnung $x$, für die bei einem Zustand $s$ des zu erzeugenden Prozeßautomaten einen Zustandsübergang $\delta_{\mathcal{A}}(s, x)$ bestimmt werden soll, geschieht, indem weitere Aktionen zu $x$ hinzugefügt werden. Sollte bei diesem Vorgehen ein Vereinigungs- oder Rekurrenzpunkt erreicht werden, kann die Konstruktion von $x$ an dieser Stelle abgebrochen werden: Wird ein $\langle a, b\rangle$-Vereinigungspunkt erreicht, so bleibt das mit $a$ bzw. $b$ beschriftete Ereignis $e$ maximal in $x$. Allerdings ist es offenbar noch möglich, Ereignisse $e'$ zu $x$ hinzuzufügen, deren Beschriftungen unabhängig von der Beschriftung von $e$ sind.

Theorem III.2.8 sichert uns nun zu, daß es eine Semiordnung $y \geqslant x$ gibt (nämlich $x_k$), so daß die maximalen Ereignisse in $y$ entweder einen Rekurrenzpunkt herstellen (Eigenschaft (b)), oder aber überhaupt keine anderen Ereignisse mehr bewirken können (Eigenschaft (a)).

Definieren wir noch

III.2.9. Definition. Für einen Zustand $s \in S_{\mathcal{T}}$ eines Spursystems $\mathcal{T}$ über $\Sigma$ ist die Relation $D_\Sigma^{\leftarrow}(s) \subseteq D_\Sigma$ als

$$a\ D_\Sigma^{\leftarrow}(s)\ b \Leftrightarrow_{\text{Def}} s \in \text{jp}_{\langle a,b\rangle}(\mathcal{T})$$

definiert, weiterhin sei

$$D_\Sigma^{\leftarrow} =_{\text{Def}} \bigcup_{s \in S_{\mathcal{T}}} D_\Sigma^{\leftarrow}(s).$$

Jede irreflexive und symmetrische Relation $B \subseteq D_\Sigma$ mit $B \supseteq D_\Sigma^{\leftarrow}$ heißt *Rückwärtskonfliktrelation* in $\mathcal{T}$. □

Damit haben wir Problem (P2) gelöst. Ist $x$ eine bei einem Zustand $s$ konzessionierte Semiordnung, fügen wir nur dann ein neues Ereignis $e$ mit einer Beschriftung $a$ zu $x$ hinzu, wenn die folgenden Bedingungen erfüllt sind

(T1) $\delta_{\mathcal{T}}(s, x) \stackrel{a}{\underset{\mathcal{T}}{\longrightarrow}}$, d. h. $a$ ist konzessioniert nach dem Schalten von $x$ bei $s$;

(T2) $F(a) = \varnothing$, wobei $F$ eine Vorwärtskonfliktrelation ist. D. h. es gibt keine Aktion in Vorwärtskonflikt zu $a$; und

```
    procedure extend(x : in out CSO(Σ); s : in out S_𝒯) is
      var A ⊆ A_Σ;
    begin
(1)   A ← addable(x, s);
(2)   while A ≠ ∅ do
(3)     select a ∈ A; x ← x ∘_Σ a; s ← δ_𝒯(s, a)
(4)     A ← addable(s, x)
(5)   od
    end extend;
```

ALGORITHMUS III.2. Prozedur *extend*.

(T3) ist $B$ eine Rückwärtskonfliktrelation und $e \in E_{x \circ_\Sigma a}$ ein Ereignis mit nicht leerer Rückwärtskonfliktmenge, so bleibt $e$ maximal in $x \circ_\Sigma a$, d. h. aus $B(\lambda_{x \circ_\Sigma a}(e)) \neq \varnothing$ folgt $e \in \max(x \circ_\Sigma a)$.

Theorem III.2.8 sichert uns zu, daß diese Prozedur letztendlich terminiert.

**Teilalgorithmen.** Wir gehen davon aus, daß eine Vorwärts- und eine Rückwärtskonfliktrelation vorab für das Spursystem $\mathcal{T}$ berechnet wurde, das Eingabe von Algorithmus III.1 ist. In Algorithmus III.1 finden die folgenden Unterroutinen Verwendung:

(a) *steps*$(s)$ gibt die Menge aller maximalen Cliquen innerhalb der Transitionsmenge $\mathrm{en}_{\mathcal{T}}(s)$ bzgl. der Relation $I(s)$ zurück.

(b) *so*$(C)$ gibt eine Semiordnung $x$ mit leerer Kausalrelation für eine Aktionsmenge $C$ zurück, d. h. ist $C = \{a_1, a_2, \ldots, a_n\}$, wird $x = \langle C, \varnothing, \mathrm{id}_C \rangle$ zurückgegeben.

(c) Die Prozedur *extend*$(x, s)$ ist als Algorithmus III.2 abgebildet. Sie erweitert eine Semiordnung $x$, die durch *so*$(C)$ berechnet wird, solange um weitere Ereignisse, bis keines der Kriterien T1 bis T3 mehr erfüllt werden kann. Simultan wird der Zustand $s$ zu $\delta_{\mathcal{T}}(s, x)$ aktualisiert.

(d) *addable*$(x, s)$ gibt eine Menge $A$ von Aktionen zurück, so daß für $x$ und $s$ die Kriterien T1, T2 und T3 für alle $a \in A$ erfüllt sind.

III.2.10. BEMERKUNG. Wir werden sehen (Bemerkung III.2.16), daß die Anzahl der bei einem Zustand $s$ schaltfähigen Schritte exponentiell in der Anzahl der bei diesem Zustand konzessionierten Aktionen sein kann. Deshalb ist bei der Auswahl eines Algorithmus für die Operation *steps* besondere Sorgfalt notwendig. Eine naive Wahl führt leicht dazu, daß Cliquen mehrfach aufgezählt werden. Ein Algorithmus, der dies vermeidet, ist das Verfahren von C. Bron und J. Kerbosch [**12, 72**]. □

III.2.11. BEMERKUNG. Zur Berechnung einer Vorwärts- bzw. Rückwärtskonfliktrelation werden wir in Abschnitt III.3 Überlegungen anstellen, wir wollen aber an dieser Stelle bereits einige Möglichkeiten zur Definition solcher Relationen diskutieren. Für

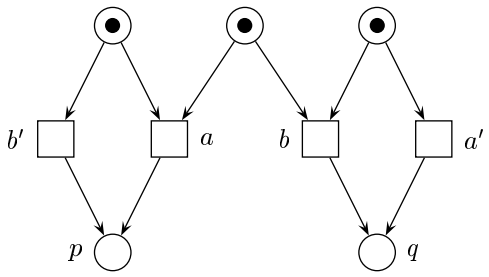



ABBILDUNG III.8. Ein Gegenbeispiel zur Festlegung (III.2.$\alpha$).

Spursysteme, deren zugehöriges Spuralphabet $\Sigma$ über eine „kleine“ Abhängigkeitsrelation verfügt (d. h. $|D_\Sigma| \ll |I_\Sigma|$), ist $F = B = D_\Sigma - \mathrm{id}_{A_\Sigma}$ sicher eine akzeptable Wahl. Allerdings ist dies für viele Systeme nicht der Fall.

Betrachten wir Netzsysteme, stellen wir etwa folgendes fest: Sind $t_1, t_2 \in T_\mathcal{N}$ Transitionen eines Netzsystems, so daß $t_1^\cdot \cap {}^\cdot t_2 \neq \varnothing$ gilt, dann sind $t_1$ und $t_2$ bei keinem Zustand $Q$ von $\mathcal{N}$ simultan konzessioniert. Entweder ist $Q \cap {}^\cdot t_2 = \varnothing$, dann kann $t_2$ erst durch das Schalten von $t_1$ konzessioniert werden, oder aber es liegt mit $Q \cap {}^\cdot t_2 \neq \varnothing$ ein Kontakt vor, dann kann $t_1$ erst schalten, nachdem $t_2$ geschaltet hat. Aber natürlich gilt $t_1 \; D_\mathcal{N} \; t_2$. Wenn wir also stark zusammenhängende Netzsysteme betrachten, wie das in der Praxis i. d. R. der Fall sein wird, existiert für jede Transition $t_1$, die bei einem Zustand $Q$ konzessioniert ist, eine weitere bei $Q$ nicht konzessionierte Transition $t_2$ mit $t_1 \; D_\mathcal{N} \; t_2$. Das bedeutet, daß $I(Q) = \mathrm{id}_{T_\mathcal{N}}$ für alle Zustände $Q$ von $\mathcal{N}$ ist: Algorithmus III.1 erzeugt für alle Transitionen $t \in \mathrm{en}_\mathcal{N}(Q)$ lediglich Einzelschritte, d. h. Algorithmus III.1 stellt eine kostspielige Art dar, das Spursystem $\mathcal{T}(\mathcal{N})$ explizit aufzubauen.

Betrachten wir kontaktfreie Netzsysteme $\mathcal{N}$ mit der häufig erfüllten Einschränkung, daß ${}^\cdot t \neq \varnothing$ und $t^\cdot \neq \varnothing$ gilt, so können wir eine Vorwärtskonfliktrelation $F$ wie folgt definieren:

$$t_1 \; F \; t_2 \Leftrightarrow_{\mathrm{Def}} {}^\cdot t_1 \cap {}^\cdot t_2 \neq \varnothing.$$

Allerdings zeigen praktische Beispiele (vgl. [**37, 38**]), daß für einen überwiegenden Anteil der Transitionen $t$ solcher Netze $|({}^\cdot t)^\cdot| > 1$ gilt; unser Grundalgorithmus stellt sich praktisch auch für kontaktfreie Netze als unbrauchbar heraus.

Der duale Ansatz zur Definition einer Rückwärtskonfliktrelation $B$ schlägt übrigens fehl: Definierten wir

$$t_1 \; B \; t_2 \Leftrightarrow_{\mathrm{Def}} t_1^\cdot \cap t_2^\cdot \neq \varnothing, \qquad \text{(III.2.}\alpha\text{)}$$

so befänden sich die beiden Transitionen $a$ und $b$ des Netzsystems in Abb III.8 nicht in $B$-Relation zueinander (obwohl allerdings $B(a) \neq \varnothing$ und $B(b) \neq \varnothing$ gilt). Der Zustand $\{p, q\}$ ist aber ein $a, b$-Vereinigungspunkt. □

**Korrektheit, Termination und Aufwand des Grundalgorithmus.** Wir zeigen nun, daß Algorithmus III.1 partiell und total korrekt ist bzgl. der Aufgabe,

einen vollständigen und konfliktvollständigen Prozeßautomaten für ein gegebenes wohlgeformtes Spursystem zu konstruieren. Weiterhin stellen wir eine — wenn auch grobe — Aufwandsbetrachtung an.

III.2.12. THEOREM. *Algorithmus III.1 terminiert für jedes wohlgeformte Spursystem $\mathcal{T}$.*

BEWEIS. Die Termination von Algorithmus III.2 folgt aus Theorem III.2.8. Da $\mathcal{T}$ jedoch nur endlich viele Zustände aufweist, folgt das Theorem. □

III.2.13. THEOREM. *Ist $\mathcal{A}$ ein Prozeßautomat, der mit Hilfe von Algorithmus III.1 für ein wohlgeformtes Spursystem $\mathcal{T}$ erzeugt wurde, so ist $\mathcal{A}$ ein Prozeßautomat zu $\mathcal{T}$.*

BEWEIS. $s_{\mathcal{A}} = s_{\mathcal{T}}$ gilt aufgrund der Zuweisung in Zeile (1) von Algorithmus III.1. $S_{\mathcal{A}} \subseteq S_{\mathcal{T}}$ gilt initial nach Ausführung von Zeile (1). Bei jedem Schleifendurchlauf wird ein $s \in S_{\mathcal{A}}$ ausgewählt (man beachte, daß $Q \subseteq S_{\mathcal{A}}$ zu jedem Zeitpunkt der Ausführung des Algorithmus gilt); nehmen wir als Induktionshypothese $s \in S_{\mathcal{T}}$ an. Es werden nun mit Hilfe der Prozeduren *steps*, *so* und *extend* Semiordnungen $x$ mit $\delta_{\mathcal{T}}(s, x) = s'$ konstruiert; dies folgt aus Korollar I.3.43. Damit gilt $s' \in S_{\mathcal{T}}$. Dies zeigt auch $s \xrightarrow[\mathcal{A}]{x} s' \Rightarrow s \xRightarrow[\mathcal{T}]{x} s'$ für alle $s, s' \in S_{\mathcal{A}}$ und $x \in X_{\mathcal{A}}$. □

III.2.14. THEOREM. *Ist $\mathcal{A}$ ein Prozeßautomat zu einem wohlgeformten Spursystem $\mathcal{T}$, der mit Hilfe von Algorithmus III.1 erzeugt wurde, so ist $\mathcal{A}$ vollständig bzgl. $\mathcal{T}$.*

BEWEIS. Zur Vereinfachung der Notation bezeichnen wir mit $\mathrm{ext}(C, s)$ für eine Clique $C \subseteq A_\Sigma$ bzgl. der Relation $I(s)$, wie sie durch die Prozedur *steps* in Zeile (4) von Algorithmus III.1 gebildet wird, die Semiordnung $x$, die durch die Ausführung der Prozedur *extend* angewendet auf $so(C)$ und einen Zustand $s \in S_{\mathcal{A}}$ konstruiert wird.

Nehmen wir ein $\boldsymbol{x}^* \in \mathbf{LSSL}(\mathcal{T}) - \mathbf{SL}(\mathcal{A})$ an und wählen wir ein Semiwort $\boldsymbol{x} \in \mathbf{LSSL}(\mathcal{T}) - \mathbf{SL}(\mathcal{A})$ so, daß $\boldsymbol{x}$ minimal bzgl. $\leqslant$ ist und $\boldsymbol{x} \leqslant \boldsymbol{x}^*$ gilt. Weiterhin sei $\boldsymbol{y} \in \mathbf{SL}(\mathcal{A})$ ein bzgl. $\leqslant$ maximales Semiwort mit $\boldsymbol{y} \leqslant \boldsymbol{x}$. Offenbar können wir dann $\boldsymbol{x}$ und $\boldsymbol{y}$ als

$$\boldsymbol{x} = \left[x^* \left[\leqslant_{x^*}^{-1} (e)\right]\right] \text{ und } \boldsymbol{y} = \left[x^* \left[<_{x^*}^{-1} (e)\right]\right]$$

für ein $e \in E_{x^*}$ schreiben: Nehmen wir $\lambda_{x^*}(e) = a$ an, erhalten wir $\boldsymbol{x} = \boldsymbol{y} \circ_\Sigma \boldsymbol{a}$. Natürlich ist $y \not\equiv \mathrm{con}_\Sigma(\chi)$ für alle Wege $\chi \in \mathbf{P}(\mathcal{A})$, da andernfalls ein Zustand $s \in S_{\mathcal{A}}$ mit $\delta_{\mathcal{T}}(s_{\mathcal{T}}, y) = s$ und $s \xrightarrow[\mathcal{T}]{a}$ existieren würde, woraus wiederum $[y \circ_\Sigma a] \in \mathbf{SL}(\mathcal{A})$ folgen würde, denn für $a$ würde bei $s$ ein Schritt $C$ mit $a \in C$ erzeugt und expandiert.

Da aber $\boldsymbol{y} \in \mathbf{SL}(\mathcal{A})$ ist, gibt es einen Weg $\chi \in \mathbf{P}(\mathcal{A})$, so daß $y \leqslant \mathrm{con}_\Sigma(\chi)$ gilt. Wir können $\chi = z_0 z_1 \ldots z_{n-1}$ so wählen, daß $y$ in Semiordnungen $y_0, y_1, \ldots, y_{n-1}$ zerlegt werden kann, so daß $y = y_0 \circ_\Sigma y_1 \circ_\Sigma \cdots \circ_\Sigma y_{n-1}$ und $y_i \leqslant z_i$ für $0 \leqslant i < n$

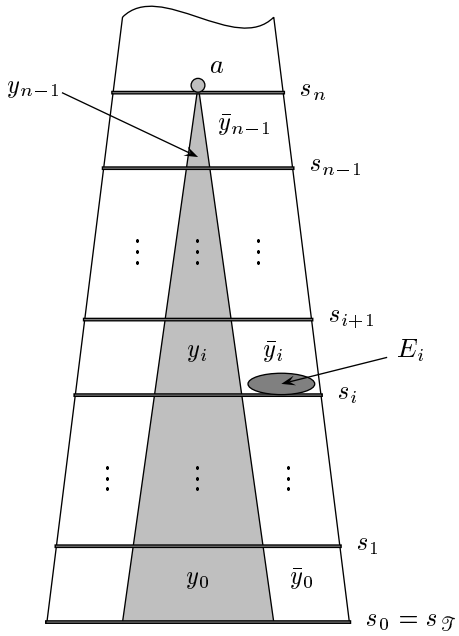



ABBILDUNG III.9. Illustration zum Beweis von Theorem III.2.13

gilt. Wenn wir mit $\bar{y}_i$ diejenige Semiordnung mit $y_i \xrightarrow{\bar{y}_i} z_i$ bezeichnen, erhalten wir $z_i = y_i \circ_\Sigma \bar{y}_i$. Setzen wir weiterhin $\bar{y} = \bar{y}_0 \circ_\Sigma \bar{y}_1 \circ_\Sigma \cdots \circ_\Sigma \bar{y}_{n-1}$, gilt $\mathrm{con}_\Sigma(\chi) = y \circ_\Sigma \bar{y}$. Außerdem gibt es natürlich Zustände $s_0, s_1, \ldots, s_n \in S_{\mathcal{A}}$ mit $s_0 = s_{\mathcal{A}}$ und

$$s_0 \xrightarrow[\mathcal{A}]{z_0} s_1 \xrightarrow[\mathcal{A}]{z_1} \cdots \xrightarrow[\mathcal{A}]{z_{n-1}} s_n$$

Abbildung III.9 illustriert die gewählten Begriffe.

Nachdem wir dieses „Koordinatensystem“ etabliert haben, können wir es verwenden, um innerhalb der Semiordnungen $\bar{y}_i$ ein Ereignis $e' \in E_{\bar{y}_i}$ mit $\lambda_{\bar{y}_i}(e') = b\ D_\Sigma\ a$ zu lokalisieren. Ein derartiges Ereignis gibt es, da $a\ I_\Sigma\ \lambda_{\bar{y}}(e')$ für alle $e' \in E_{\bar{y}}$ einerseits nach Lemma I.3.14 $s_{\mathcal{A}} \xrightarrow[\mathcal{T}]{\mathrm{con}_\Sigma(\chi)} s_n \xrightarrow[\mathcal{T}]{a}$ implizieren, andererseits jedoch $x \leqslant \mathrm{con}_\Sigma(\chi) \circ_\Sigma a$ und damit $\boldsymbol{x} \in \mathbf{SL}(\mathcal{A})$ gelten würde.

Sei $E =_{\mathrm{Def}} \{e \in E_{\bar{y}} : a\ D_\Sigma\ \lambda_{\bar{y}}(e)\}$ und $E_i =_{\mathrm{Def}} E \cap E_{\bar{y}_i}$ für $0 \leqslant i < n$; $E$ ist dann nicht leer. Offenbar gilt $a\ D_\Sigma^{\rightarrow}\ b = \lambda_{\bar{y}}(e')$ für alle $e' \in E_i$, der notwendige Zustand $s \in S_{\mathcal{T}}$ mit $s \xrightarrow[\mathcal{T}]{a}$ und $s \xrightarrow[\mathcal{T}]{b}$ ist

$$s = \delta_{\mathcal{T}}^{\mathrm{con}_\Sigma(\chi)} \left( s_{\mathcal{T}}, H_y^{\mathrm{con}_\Sigma(\chi)}(E_y) \cup <_{\mathrm{con}_\Sigma(\chi)}^{-1} \left( H_{\bar{y}}^{\mathrm{con}_\Sigma(\chi)}(E_i) \right) \right).$$

Sei $i < n$ der kleinste Index, so daß $E_i \neq \varnothing$ gilt. Man beachte, daß dann

$$s_i \xrightarrow[\mathcal{T}]{y_i \circ_\Sigma y_{i+1} \circ_\Sigma \cdots \circ_\Sigma y_{n-1}}$$

gilt. Ausgehend vom Zustand $s_i$ zeigen wir nun, daß wiederholte Aufrufe von Algorithmus III.2 eine Semiordnung $\tilde{z}$ mit

$$y_i \circ_\Sigma y_{i+1} \circ_\Sigma \cdots \circ_\Sigma y_{n-1} \circ_\Sigma a \leqslant \tilde{z}$$

erzeugen. Für den Zustand $s_i$ haben wir die folgenden Fälle zu unterscheiden:

*Fall 1.* Für alle $e \in \min(y_i)$ gilt $F(\lambda_{y_i}(e)) \neq \varnothing \Rightarrow F(\lambda_{y_i}(e)) \subseteq \mathrm{en}_{\mathcal{T}}(s_i)$, wobei $F$ die Vorwärtskonfliktrelation ist, die bei der Ausführung von Algorithmus III.1 verwendet wird. In diesem Fall gibt es eine maximale Clique $C \subseteq A_\Sigma$ bzgl. der Relation $I(s)$ mit $\lambda_{y_i}(\min(y_i)) \subseteq C$; es gilt

$$D^{\rightarrow}_\Sigma(c) \neq \varnothing \Rightarrow D^{\rightarrow}_\Sigma(c) \subseteq \mathrm{en}_{\mathcal{T}}(s_i) \qquad \text{(III.2.}\beta\text{)}$$

für alle $c \in C$, da $F \supseteq D^{\rightarrow}_\Sigma$ ist. Ist (i) $a \in C$, so ist $C \cap \lambda_{y_i}(E_i) = \varnothing$, ist andererseits (ii) $a \notin \mathrm{en}_{\mathcal{T}}(s_i)$, gilt ebenfalls $C \cap \lambda_{y_i}(E_i) = \varnothing$, da die Elemente $b \in \lambda_{y_i}(E_i)$ die Bedingung (III.2.$\beta$) nicht erfüllen. Schließlich ist (iii) der Fall $a \in \mathrm{en}_{\mathcal{T}}(s_i)$ und $a \notin C$ zu behandeln. Dies ist jedoch genau die im Teil II der Diskussion von Problem (P1) dargestellte Situation: Mit den dort angestellten Überlegungen folgt $\lambda_{y_1}(E_i) \cap (C - \lambda_{y_i}(\min(y_i))) = \varnothing$.

Wird nun die Semiordnung $\tilde{z}_i =_{\mathrm{Def}} \mathrm{ext}(C, \delta_{\mathcal{T}}(s_i, so(C)))$ gebildet, so geschieht dies indem lediglich Aktionen $c'$ mit $F(c') = \varnothing$ zu $so(C)$ hinzugefügt werden. Weiterhin folgt aufgrund der Wahl des Terminationskriteriums für *extend*, daß $y_i \leqslant \tilde{z}_i$ sowie weiterhin $e \in \max(y_i) \Rightarrow H^{\tilde{z}_i}_{y_i}(e) \in \max(\tilde{z}_i)$. Der wesentliche Punkt jedoch ist, daß für die Semiordnung $\tilde{y}_i =_{\mathrm{Def}} \tilde{z}_i \left[E_{\tilde{z}_i} - H^{\tilde{z}_i}_{y_i}(E_{y_i})\right]$ gilt: $a \; I_\Sigma \; \lambda_{\tilde{y}_i}(e)$ für alle $e \in E_{\tilde{y}_i}$.

*Fall 2.* Es gibt ein $e \in \min(y_i)$, so daß $F(\lambda_{y_i}(e)) \neq \varnothing$, jedoch auch $F(\lambda_{y_i}(e)) \not\subseteq \mathrm{en}_{\mathcal{T}}(s_i)$ gilt. In diesem Fall wird bei $s_i$ durch die Prozedur *steps* in Algorithmus III.1 ein Einzelschritt $C = \{\lambda_{y_i}(e)\}$ konstruiert, mit dem die Prozedur *extend* parametrisiert wird. Mit einer Argumentation analog zu Fall 1 schließen wir, daß für $\tilde{z}_i =_{\mathrm{Def}} \mathrm{ext}(C, \delta_{\mathcal{T}}(s_i, so(C)))$ gilt: $y_i \leqslant \tilde{z}_i$, $e \in \max(y_i) \Rightarrow H^{\tilde{z}_i}_{y_i}(e) \in \max(\tilde{z}_i)$ und $a \; I_\Sigma \; \lambda_{\tilde{y}_i}(e)$ für alle $e \in E_{\tilde{y}_i}$, wobei $\tilde{y}_i$ wie eben durch $\tilde{y}_i =_{\mathrm{Def}} \tilde{z}_i \left[E_{\tilde{z}_i} - H^{\tilde{z}_i}_{y_i}(E_{y_i})\right]$ definiert ist.

Für beide Fälle setzen wir $\tilde{s}_{i+1} =_{\mathrm{Def}} \delta_{\mathcal{A}}(s_i, \tilde{z}_i)$. Wir zeigen nun, daß bei diesem Zustand eine Semiordnung $\tilde{z}_{i+1}$ mit $y_{i+1} \leqslant \tilde{z}_{i+1}$, $e \in \max(y_{i+1}) \Rightarrow H^{\tilde{z}_{i+1}}_{y_{i+1}}(e) \in \max(\tilde{z}_{i+1})$ und $a \; I_\Sigma \; \lambda_{\tilde{y}_{i+1}}(e)$ für alle $e \in E_{\tilde{y}_{i+1}}$ konstruiert wird, wobei $\tilde{y}_{i+1} =_{\mathrm{Def}} \tilde{z}_{i+1}\left[E_{\tilde{z}_{i+1}} - H^{\tilde{z}_{i+1}}_{y_{i+1}}(E_{y_{i+1}})\right]$ ist.

Wir können nun nicht voraussetzen, daß die Menge

$$\tilde{E} =_{\mathrm{Def}} \{b \in A_\Sigma : b \in \mathrm{en}_{\mathcal{T}}(\tilde{s}_{i+1}) \;\&\; a \; D_\Sigma \; b\}$$

nicht leer ist. Nehmen wir dies aber an, können wir analog zu Fall 1 und Fall 2 argumentieren.

*Fall 3.* Ist hingegen $\tilde{E} = \varnothing$ so gilt $C \cap \tilde{E} = \varnothing$, wobei $C$ eine maximale Clique von bei $\tilde{s}_{i+1}$ schaltfähigen Aktionen bzgl. der Relation $I(\tilde{s}_{i+1})$ ist; dieser Fall kann also analog zu Fall 1 behandelt werden.

Allgemein können wir also für alle $y_j$, $i \leqslant j < n$ eine Transition $\delta_{\mathscr{A}}(\tilde{s}_j, \tilde{z}_j) = \tilde{s}_{j+1}$ finden, so daß $y_j \leqslant \tilde{z}_j$ gilt, und keine der Aktionen, die als Ereignisse in $\tilde{z}_j \left[E_{\tilde{z}_j} - H_{y_j}^{\tilde{z}_j}(E_{y_j})\right]$ vorkommen, abhängig von $a$ ist. Wir verzichten darauf, die Induktion explizit vorzunehmen, da sie notwendigerweise sehr technisch ausfallen würde und die benötigte Argumentation bereits klar sein sollte.

Nun gilt aber, daß $a \in \mathrm{en}_{\mathscr{T}}(\tilde{s}_n)$ ist: Bei diesem Zustand wird ein Schritt $C$ mit $a \in C$ konstruiert, der zu einer Semiordnung $\tilde{z}_n$ erweitert wird, die ein minimales, mit $a$ beschriftetes Ereignis enthält. Damit gilt jedoch

$$y \circ_\Sigma a \leqslant z_0 \circ_\Sigma z_1 \circ_\Sigma \cdots \circ_\Sigma z_{i-1} \circ_\Sigma \tilde{z}_i \circ_\Sigma \cdots \circ_\Sigma \tilde{z}_n,$$

d. h. $[y \circ_\Sigma a] = \boldsymbol{x} \in \mathbf{SL}(\mathscr{A})$. □

III.2.15. Theorem. *Das Theorem gilt unter der Annahme, daß die folgenden Operationen in polynomieller Zeit ausgeführt werden können:*

(a) *Die Auswertung der Ausdrücke $a\ F\ b$ bzw. $a\ B\ b$ sowie $\mathrm{per}(a) = \varnothing$, wobei $F$ und $B$ die bei der Ausführung von Algorithmus III.1 verwendete Vorwärts- und Rückwärtskonfliktrelation sind.*

(b) *Die Auswertung von $a \in \mathrm{en}_{\mathscr{T}}(s)$ für einen Zustand $s$ eines Spursystems $\mathscr{T}$ über $\Sigma$ und einer Aktion $a \in A_\Sigma$.*

(c) *Die Berechnung des Zustands $\delta_{\mathscr{T}}(s, a)$ für einen Zustand $s$ eines Spursystems $\mathscr{T}$ über $\Sigma$ und einer Aktion $a \in A_\Sigma$.*

*Der Zeitaufwand für die Konstruktion eines Prozeßautomaten $\mathscr{A}$ zu einem Spursystem $\mathscr{T}$ über $\Sigma$ durch Algorithmus III.1 ist von der Ordnung $O\left(|S_{\mathscr{T}}| \cdot \left(2^k \cdot |A_\Sigma|^c\right)\right)$ für eine Konstante $c > 0$, wobei $k$ die maximale Größe einer Clique $C \subseteq A_\Sigma$ bzgl. der Relation $D_\Sigma$ ist.*

Beweis. Ist $D_\Sigma = A_\Sigma \times A_\Sigma$, so entspricht der durch Algorithmus III.1 konstruierte Prozeßautomat dem Spursystem $\mathscr{T}$ in dem Sinne, daß es sich bei den Semiordnungen, für die Transitionen in $\mathscr{A}$ definiert sind, um Buchstaben handelt. Das erklärt den ersten Faktor $|S_{\mathscr{T}}|$.

$2^k$ errechnet sich wie folgt: Für jeden Zustand $s \in S_{\mathscr{A}}$ von $\mathscr{A}$ werden durch die Prozedur *steps* höchstens $d \cdot 2^k$ Schritte für eine Konstante $d > 0$ konstruiert (Beispiele, in denen die Gleichheit gilt, können leicht konstruiert werden, wie wir in der folgenden Bemerkung zeigen werden). Jeder dieser Schritte wird höchstens durch $|A_\Sigma|$ Ereignisse zu einer Semiordnung erweitert; das Hinzufügen eines Ereignisses zu einer Semiordnung $x$ kann in polynomieller Zeit geschehen — diese Operation ist wenigstens von der Ordnung $O(|E_x|^2 \cdot |A_\Sigma|)$. Damit ist der Zeitaufwand, der zur vollständigen Bearbeitung eines Zustandes benötig wird, von der Ordnung $O(2^k \cdot |A_\Sigma|^c)$. □

III.2.16. Bemerkung. Die Existens von $d \cdot 2^k$ Schritten mit

$$k = \max\{|C| : C \text{ max. Clique bzgl. } I(s) \text{ in } A_\Sigma\}$$

wird durch das folgende Beispiel belegt: Definieren wir eine Komponente $\mathscr{C}$ eines Netzsystems $\mathcal{N}$ als ein Teilnetz, das zwei Transitionen $t_1$ und $t_2$ und einen Platz $p$ enthält, so daß

$$F_{\mathscr{C}} = \{\langle p, t_1\rangle, \langle p, t_2\rangle\}$$

und $p \in Q_{\mathscr{C}}$ gilt. Bei $Q_{\mathscr{C}}$ existieren $2^1$ konzessionierte Schritte, nämlich $\{t_1\}$ und $\{t_2\}$. Sei nun $\mathcal{N}_n$ ein Netzsystem, das aus $n$ solcher Komponenten besteht, wobei zwischen den einzelnen Komponenten kein Zusammenhang besteht. Ein einfaches induktives Argument über $n$ zeigt nun, daß $\mathcal{N}_n$ $2^n$ konzessionierter Schritte bei $Q_{\mathcal{N}_n}$ aufweist. □

III.2.17. Theorem. *Der durch Algorithmus III.1 für ein wohlgeformtes Spursystem $\mathcal{T}$ über $\Sigma$ erzeugte Prozeßautomat ist konfliktvollständig.*

Beweis. Das Theorem folgt sofort aus der Verwendung einer Vorwärtskonfliktrelation $F$ und einer Rückwärtskonfliktrelation $B$ zur Initiierung und Terminierung der Prozedur *extend*. □

## III.3. Verbesserungen des Grundalgorithmus

Wir befassen uns in diesem Abschnitt mit verschiedenen Verbesserungen des Laufzeit- und Speicherplatzverhaltens des Grundalgorithmus III.1. Eine erste Verbesserung des Speicherverhaltens ist sehr einfach: Nach Erzeugung eines Prozeßautomaten können bestimmte Zustände als redundant identifiziert und entfernt werden.

Die zweite Verbesserung beruht auf der Beobachtung, daß die Betrachtung bestimmter Aktionen, die bei einem Zustand $s$ eines Spursystems $\mathcal{T}$ konzessioniert sind, der während der Konstruktion eines Prozeßautomaten auftritt, auf einen späteren Zeitpunkt verschoben werden kann. Dies führt zu einer Reformulierung des Grundalgorithmus.

Schließlich verwenden wir den in Abschnitt II.1 erarbeiteten Komponentenbegriff, um mit Hilfe struktureller Analysen geeignetere Konfliktrelationen $F$ und $B$ als $D_\Sigma - \mathrm{id}_{A_\Sigma}$ zu bestimmen. Zu diesem Zweck werden auch Fallen definiert, die in Verbindung mit Komponenten eine weitere Verbesserung der Wahl der Relationen $F$ und $B$ bewirken können.

**Entfernung redundanter Zustände.** Sei $\mathcal{T}$ ein wohlgeformtes Spursystem über $\Sigma$. Wie leicht einzusehen ist, mag ein durch Algorithmus III.3 konstruierter Prozeßautomat $\mathscr{A}$ Zustände $s \in S_{\mathscr{A}}$ aufweisen, so daß es genau einen Zustand $s_1 \in S_{\mathscr{A}}$ und genau einen Zustand $s_2 \in S_{\mathscr{A}}$ gibt, für die die Zustandsübergänge $s_1 \xrightarrow[\mathscr{A}]{x} s \xrightarrow[\mathscr{A}]{y} s_2$ für $x, y \in X_{\mathscr{A}}$ definiert sind. Solche Zustände $s$ sind offenbar redundant

und können zusammen mit beiden inzidenten Kanten nach erfolgter Konstruktion aus dem Prozeßautomaten entfernt werden. Stattdessen wird ein neuer Übergang $\delta_A(s_1, x \circ_\Sigma y) =_{\text{Def}} s_2$ definiert.

---

**algorithm** *generate* **is**
  **input** $\mathcal{T}$, ein wohlgeformtes Spursystem über $\Sigma$;
  **output** $\mathcal{A}$, ein Prozeßautomat zu $\mathcal{T}$;
  **local variables** $Q$ : **stack of** $S_{\mathcal{T}}$;
    $T$ : **set of** $S_{\mathcal{T}} \times S_{\mathcal{T}} \leftarrow \varnothing$; *loop* : **bool**;
    $s, s'$ : $S_{\mathcal{T}}$; $x$ : **CSO**$(\Sigma)$; $U, V$ : **set of set of** $A_\Sigma$;
**begin**
(1) $s_{\mathcal{A}} \leftarrow s_{\mathcal{T}}$; $S_{\mathcal{A}} \leftarrow \{s_{\mathcal{A}}\}$; $X_{\mathcal{A}} \leftarrow \varnothing$; $\delta_{\mathcal{A}} \leftarrow \varnothing$; $push(Q, s_{\mathcal{A}})$;
(2) **while** $\neg empty(Q)$ **do**
(3)  $s \leftarrow top(Q)$; $pop(Q)$; $loop \leftarrow$ **false**;
(4)  $V \leftarrow ignore(s)$; $U \leftarrow steps\left(\text{en}_{\mathcal{T}}(s) - \bigcup_{C \in V} C\right)$;
(5)  **if** $U \neq \varnothing$ **then**
(6)  **foreach** $C \in U$ **do**
(7)  $x \leftarrow so(C)$; $s' \leftarrow \delta_{\mathcal{T}}(s, x)$; $extend(x, s')$;
(8)  **if** $s' \notin S_{\mathcal{A}}$ **then**
(9)  $push(Q, s')$; $S_{\mathcal{A}} \leftarrow S_{\mathcal{A}} \cup \{s'\}$; $T \leftarrow T \cup \{\langle s', s\rangle\}$
(10)  **else**
(11)  $loop \leftarrow loop \vee s\ T^*\ s'$
(12)  **fi**;
(13)  $X_{\mathcal{A}} \leftarrow X_{\mathcal{A}} \cup \{x\}$; $\delta_{\mathcal{A}}(s, x) \leftarrow s'$
(14)  **od**
(15)  **fi**;
(16)  **if** $loop \vee U = \varnothing$ **then**
(17)  **foreach** $C \in V$ **do**
(18)  $x \leftarrow so(C)$; $s' \leftarrow \delta_{\mathcal{T}}(s, x)$; $extend(x, s')$;
(19)  **if** $s' \notin S_{\mathcal{A}}$ **then**
(20)  $push(Q, s')$; $S_{\mathcal{A}} \leftarrow S_{\mathcal{A}} \cup \{s'\}$; $T \leftarrow T \cup \{\langle s', s\rangle\}$
(21)  **fi**;
(22)  $X_{\mathcal{A}} \leftarrow X_{\mathcal{A}} \cup \{x\}$; $\delta_{\mathcal{A}}(s, x) \leftarrow s'$
(23)  **od**
(24)  **fi**
(25) **od**
**end** *generate*.

ALGORITHMUS III.3. Reformulierung des Grundalgorithmus III.1.

**Verschieben der Betrachtung konzessionierter Transitionen.** Sei $s$ ein Zustand des Spursystems $\mathcal{T}$ über einem Spuralphabet $\Sigma$, der während der Konstruktion eines Prozeßautomaten zu $\mathcal{T}$ auftritt, und sei $F$ eine Vorwärtskonfliktrelation zu $\mathcal{T}$. Wie wir in Abschnitt III.2 dargestellt haben, gilt $I(s)(a) = \{a\}$ für solche Transitionen $a \in A_\Sigma$ mit $F(a) \not\subseteq \mathrm{en}_{\mathcal{T}}(s)$, d. h. durch Algorithmus III.1 wird ein Einzelschritt $C = \{a\}$ als Grundlage für die Konstruktion einer bei $s$ schaltfähigen Semiordnung erzeugt. Nehmen wir weiter $F(a) \cap \mathrm{en}_{\mathcal{T}}(s) = \varnothing$ an. Sei $C' \subset \mathrm{en}_{\mathcal{T}}(s)$ ein weiterer Schritt mit $a \notin C'$ und sei $x \equiv \langle C', \varnothing, \mathrm{id}_{C'} \rangle$ die mit $C'$ assoziierte Semiordnung. Die Prozedur *extend* fügt zu $x$ ausschließlich Transitionen mit leerer Vorwärtskonfliktmenge hinzu, d. h. ist $x$ nach Termination von *extend* zu $y$ erweitert worden, *ist $a$ weiterhin schaltfähig nach dem Feuern von $y$ bei $s$*. Es ist deshalb nicht notwendig, $a$ bei $s$ zu beachten. Die Betrachtung von $a$ kann verschoben werden, bis ein Zustand erzeugt wird, bei dem $a$ einem Schritt mit mehr als einem Element zugeordnet werden kann.

Diese Regel hat zwei Ausnahmen:

(a) Jede Transition $b \in \mathrm{en}_{\mathcal{T}}(s)$ hat die oben genannte Eigenschaft: $F(b) \not\subseteq \mathrm{en}_{\mathcal{T}}(s) \Rightarrow F(b) \cap \mathrm{en}_{\mathcal{T}}(s) = \varnothing$ oder

(b) das Hinzufügen einer Transition $\delta_{\mathcal{T}}(s, y) =_{\mathrm{Def}} s'$ führt zu einem Kreis in dem zu konstruierenden Prozeßautomaten, d. h. der Zustand $s'$ wurde bereits erzeugt und $s$ ist über eine Folge von Transitionen des Prozeßautomaten von $s'$ aus erreichbar. In diesem Fall würde $a$ fortwährend ignoriert werden.

*Reformulierung des Grundalgorithmus.* Die folgenden Datenstrukturen finden in der Reformulierung III.3 des Grundalgorithmus Verwendung:

(a) $Q$, ein Keller, enthält solche Zustände von $\mathcal{T}$, die noch nicht vollständig betrachtet wurden.

(b) $T$ ist eine Relation, die den sog. *spannenden Baum* des erzeugten Prozeßautomaten $\mathcal{A}$ repräsentiert. $T$ dient zum Aufspüren von Kreisen in $\mathcal{A}$. Wir kommen im nächsten Teilabschnitt nach einer genaueren Analyse der durch Algorithmus III.3 vorgenommenen Tiefensuche in $\mathcal{A}$ darauf zurück.

(c) $U$ und $V$ sind Mengen von Aktionsmengen. $U$ enthält solche Schritte, die bei einem betrachteten Zustand $s$ zu einer Semiordnung erweitert werden können, $V$ enthält Einzelschritte, die möglicherweise bei $s$ noch nicht betrachtet werden müssen. Die Schleife (6) – (14) behandelt Schritte in $U$, Einzelschritte aus $V$ werden in der Schleife (17) – (23) bearbeitet.

Weiterhin wird neben den in Abschnitt III.2 diskutierten Prozeduren der Teilalgorithmus *ignore* verwendet:

$$\mathit{ignore}(s) \;=_{\mathrm{Def}}\; \big\{\{a\} \in \mathcal{P}(A_\Sigma) : a \in \mathrm{en}_{\mathcal{T}}(s) \;\&\; F(a) \not\subseteq \mathrm{en}_{\mathcal{T}}(s) \Rightarrow F(a) \cap \mathrm{en}_{\mathcal{T}}(s) = \varnothing\big\}$$

gibt für einen Zustand $s \in S_{\mathcal{T}}$ die Menge derjenigen Einzelschritte von Aktionen aus $\mathrm{en}_{\mathcal{T}}(s)$ zurück, die möglicherweise noch nicht betrachtet werden müssen. Die

Prozedur *steps* wird nun mit einer Aktionsmenge $A \subseteq \mathrm{en}_{\mathcal{J}}(s)$ anstatt mit dem Zustand $s$ selbst parametrisiert: $steps(A)$ gibt die Menge der maximalen Schritte in $A$ bzgl. der Relation $I(s)$ zurück.

*Analyse der Tiefensuche.* Bevor wir die Korrektheit von Algorithmus III.3 nachweisen können, benötigen wir weitere Begriffe. Zur Argumentation über Kreise in einem gerichteten Graphen und ihrer Auffindung mit Hilfe einer Tiefensuche, wie sie durch Algorithmus III.3 vorgenommen wird, ist eine genaue Analyse der Tiefensuche nützlich. Eine ausführlichere Diskussion findet sich in [**54**].

III.3.1. Definition. Ist $\mathcal{A}$ ein Prozeßautomat zu einem Spursystem $\mathcal{T}$ über $\Sigma$, so bezeichnen wir die Struktur $\mathcal{G}(\mathcal{A}) =_{\mathrm{Def}} \langle S_{\mathcal{A}}, R_{\mathcal{A}}, s_{\mathcal{A}} \rangle$ als *Graphen* zu $\mathcal{A}$, wobei die Relation $R_{\mathcal{A}} \subseteq S_{\mathcal{A}} \times S_{\mathcal{A}}$ als

$$s_1 \; R_{\mathcal{A}} \; s_2 \Leftrightarrow_{\mathrm{Def}} \exists x \in X_{\mathcal{A}} \left( s_1 \xrightarrow[\mathcal{A}]{x} s_2 \right)$$

definiert ist. □

Bei der oben definierten Struktur handelt es sich um einen Graphen, der über eine *Wurzel* verfügt, d. h. einen Knoten ohne einlaufende Kanten, von dem aus jeder andere Knoten des Graphen über einen Kantenzug erreichbar ist. Die folgende Definition beschreibt, was wir in diesem Abschnitt unter einem Graphen verstehen wollen. Teilgraphen von Graphen umfassen vollständig deren Knotenmenge, weisen aber u. U. nur eine Teilmenge ihrer Kantenmenge auf. Dabei ist wichtig, daß der Wurzelknoten eines Teilgraphen mit dem des zugrunde liegenden Graphen übereinstimmt.

III.3.2. Definition. Allgemein sprechen wir von einer Struktur $\mathcal{G} = \langle S, R, s \rangle$ als von einem *endlichen*, *gerichteten* und *verankerten Graphen* oder kurz als von einem *Graphen*, wenn $S$ eine endliche Menge von *Knoten*, $R \subseteq S \times S$ eine *Kantenrelation* und $s \in S$ ein *Startknoten* bzw. eine *Wurzel* ist, so daß $s \; R^* \; s'$ für alle $s' \in S$ gilt.

Ist $\mathcal{G}$ ein Graph, so nennen wir einen Graphen $\mathcal{F}$ einen *Teilgraphen* von $\mathcal{G}$, wenn $S_{\mathcal{F}} = S_{\mathcal{G}}$, $s_{\mathcal{F}} = s_{\mathcal{G}}$ und $R_{\mathcal{F}} \subseteq R_{\mathcal{G}}$ gilt. □

Ein Baum ist ein Graph, dessen Knoten mit Ausnahme des Wurzelknotens genau eine einlaufende Kante aufweisen, wobei keine Kante in die Wurzel einläuft. Teilgraphen von Graphen, die Bäume sind, werden als spannende Bäume bezeichnet. Für jeden Graphen $\mathcal{G}$ im Sinne von Definition III.3.2 existiert wenigstens ein spannender Baum. Würden wir die Voraussetzung der Existenz einer Wurzel aufgeben, müßten wir von einem *spannenden Wald* reden, d. h. einer Menge von Bäumen mit paarweise disjunkten Knotenmengen, deren (komponentenweise) Vereinigung einen Teilgraph von $\mathcal{G}$ ergäbe.

III.3.3. Definition. Ein *Baum* ist ein Graph $\mathcal{G}$, so daß

$$\left| R_{\mathcal{G}}^{-1}(s) \right| \leqslant 1 \;\&\; \left( \left| R_{\mathcal{G}}^{-1}(s) \right| < 1 \Rightarrow s = s_{\mathcal{G}} \right)$$

```
algorithm dfs is
    input 𝒢, ein Graph;
    output num, cnum : array S_𝒢 of ℕ, Nummerierungen der Knoten von 𝒢;
           R_𝒢^T, R_𝒢^F, R_𝒢^B, R_𝒢^C : set of R_𝒢 ← ∅, eine Partitionierung von R_𝒢;
    local variables i, c : ℕ ← 0; S : set of S_𝒢;
    local procedures
      procedure traverse(in s : S_𝒢) is
      begin
(1')    foreach s' ∈ S_𝒢 : s R_𝒢 s' do
(2')      if s' ∉ S then
(3')        S ← S ∪ {s'}; R_𝒢^T ← R_𝒢^T ∪ {⟨s, s'⟩};
(4')        num(s') ← i; i ← i + 1;
(5')        traverse(s');
(6')        cnum(s') ← c; c ← c + 1;
(7')      else
(8')        if s R_𝒢^T* s' then R_𝒢^F ← R_𝒢^F ∪ {⟨s, s'⟩}
(9')        elsif s' R_𝒢^T* s then R_𝒢^B ← R_𝒢^B ∪ {⟨s, s'⟩}
(10')       else R_𝒢^C ← R_𝒢^C ∪ {⟨s, s'⟩}
(11')       fi;
(12')     fi
(13')   od
      end traverse
begin
(1)  S ← {s_𝒯};
(2)  num(s_𝒢) ← i; i ← i + 1; traverse(s_𝒢); cnum(s_𝒢) ← c;
end dfs.
```

Algorithmus III.4. Tiefensuche.

für alle $s \in S_{\mathcal{G}}$ gilt. Ist ein Teilgraph $\mathcal{F}$ von $\mathcal{G}$ ein Baum, so heißt $\mathcal{F}$ *spannender Baum* von $\mathcal{G}$. □

Bekanntermaßen definiert eine Tiefensuche innerhalb eines Graphen einen spannenden Baum dieses Graphen. Zur Illustration dient Algorithmus III.4. Neben den Knotennummerierungen *num* und *cnum*, die der Reihenfolge entsprechen, in der die Knoten des Eingabegraphen $\mathcal{G}$ besucht werden bzw. ihr Besuch abgeschlossen ist (d. h. alle Nachfolger eines Knotens wurden besucht), wird eine Partitionierung der Kantenmenge von $\mathcal{G}$ in die Mengen $R_{\mathcal{G}}^{\mathrm{T}}$ (Baumkanten), $R_{\mathcal{G}}^{\mathrm{F}}$ (Vorwärtskanten), $R_{\mathcal{G}}^{\mathrm{B}}$ (Rückwärtskanten) und $R_{\mathcal{G}}^{\mathrm{C}}$ (Querkanten) berechnet. I. E.:

(a) $\langle s, s' \rangle$ ist eine *Baumkante*, wenn sie zu dem durch Algorithmus III.4 bestimmten spannenden Baum $\mathcal{F} = \langle V_{\mathcal{G}}, R_{\mathcal{G}}^{\mathrm{T}}, v_{\mathcal{G}} \rangle$ von $\mathcal{G}$ gehört.

(b) $\langle s, s' \rangle$ ist eine *Vorwärtskante*, wenn $s'$ von $s$ über eine Folge von wenigstens zwei Baumkanten erreichbar ist;

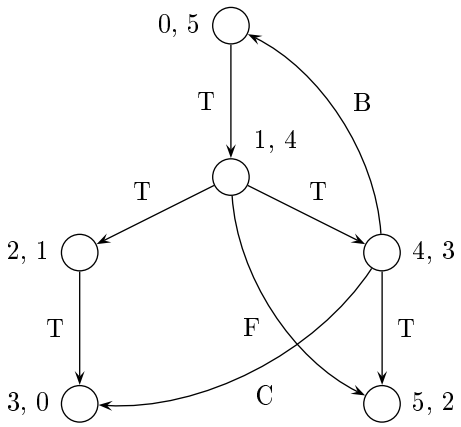



ABBILDUNG III.10. Kantenpartitionierung durch Tiefensuche.

(c) $\langle s, s' \rangle$ ist eine *Rückwärtskante*, wenn $s$ von $s'$ aus über eine Folge von Baumkanten erreichbar ist. $\langle s, s' \rangle$ schließt also einen Kreis in $\mathcal{G}$.
(d) $\langle s, s' \rangle$ ist eine *Querkante*, wenn $s'$ zum Zeitpunkt des Besuchs von $s$ innerhalb eines bereits vollständig traversierten Unterbaums von $\mathcal{F}$ liegt. Querkanten schließen keine Kreise.

III.3.4. BEISPIEL. Abbildung III.10 zeigt einen Graphen, dessen Kanten entsprechend der durch eine bestimmte Besuchsreihenfolge definierten Partitionierung bezeichnet sind. Die Knoten $s$ des Graphen sind mit Paaren $num(s), cnum(s)$ beschriftet. □

Algorithmus III.4 verwendet zur Bestimmung von Vorwärts- und Rückwärtskanten die reflexiv-transitive Hülle über $R^{\mathrm{T}}_{\mathcal{G}}$. Man kann sich leicht überlegen, daß hier der zum Zeitpunkt dieser Berechnung jeweils bereits bestimmte Anteil der vollständigen Relation $R^{\mathrm{T}}_{\mathcal{G}}$ ausreichend ist.

III.3.5. LEMMA. *Für die durch Algorithmus III.4 für einen Graphen $\mathcal{G}$ bestimmten Knotennummerierungen und die Kantenmengen $R^{\mathrm{T}}_{\mathcal{G}}$, $R^{\mathrm{F}}_{\mathcal{G}}$, $R^{\mathrm{B}}_{\mathcal{G}}$ und $R^{\mathrm{C}}_{\mathcal{G}}$ gilt:*

(a) $R_{\mathcal{G}} = R^{\mathrm{T}}_{\mathcal{G}} \cup R^{\mathrm{F}}_{\mathcal{G}} \cup R^{\mathrm{B}}_{\mathcal{G}} \cup R^{\mathrm{C}}_{\mathcal{G}}$, *weiterhin sind* $R^{\mathrm{T}}_{\mathcal{G}}$, $R^{\mathrm{F}}_{\mathcal{G}}$, $R^{\mathrm{B}}_{\mathcal{G}}$ *und* $R^{\mathrm{C}}_{\mathcal{G}}$ *paarweise disjunkt.*
(b) $s\ R^{\mathrm{T}*}_{\mathcal{G}}\ s' \Leftrightarrow num(s) \leqslant num(s')\ \&\ cnum(s) \geqslant cnum(s')$.
(c) $s\ R^{\mathrm{T}}_{\mathcal{G}}\ s' \Rightarrow num(s) < num(s')\ \&\ cnum(s) > cnum(s')$.
(d) $s\ R^{\mathrm{F}}_{\mathcal{G}}\ s' \Rightarrow num(s) < num(s')\ \&\ cnum(s) > cnum(s')$.
(e) $s\ R^{\mathrm{B}}_{\mathcal{G}}\ s' \Rightarrow num(s) > num(s')\ \&\ cnum(s) < cnum(s')$.
(f) $s\ R^{\mathrm{C}}_{\mathcal{G}}\ s' \Rightarrow num(s) > num(s')\ \&\ cnum(s) > cnum(s')$.

BEWEIS. Einfache Folgerungen aus [**54**], Lemma 2 in Abschnitt IV.5. □

Es ist wichtig, daß es keine Kanten $\langle s, s'\rangle$ gibt, so daß $s'$ in einem zum Zeitpunkt des Besuchs von $s$ noch unbesuchten Unterbaum des erzeugten spannenden Baums liegt und $\langle s, s'\rangle$ keine Baumkante ist (man stelle sich in dem Graphen in Abbildung III.10 etwa eine Kante $3,0 \longrightarrow 4,3$ vor): Eine entsprechende fünfte Kantenrelation $R$, die dann

$$s\ R\ s'\ \&\ s\ R_{\mathcal{G}}\ s' \Rightarrow num(s) < num(s')\ \&\ cnum(s) < cnum(s').$$

erfüllen würde, wird zur Bildung einer Partitionierung von $R_{\mathcal{G}}$ nicht benötigt.

Offenbar definiert jede Traversierung eines Graphen einen spannenden Baum dieses Graphen wie auch die zugehörigen Abbildungen *num* und *cnum*. Gehen wir deshalb von der Reihenfolge aus, in der die Zustände eines Prozeßautomaten durch Algorithmus III.3 erzeugt werden, können wir diesen Zuständen $s$ die Nummern $num(s)$ und $cnum(s)$ zuordnen. Weiterhin können wir von den Zustandsübergängen $s \xrightarrow[\mathcal{A}]{x} s'$, die während der Prozeßautomatenerzeugung bestimmt werden, als von Baumkanten, Vorwärtskanten, Rückwärtskanten bzw. Querkanten reden.

*Korrektheit von Algorithmus III.3.* Nach dieser Vorbereitung können wir nun den Korrektheisbeweis für den verbesserten Grundalgorithmus vorlegen. Unsere Überlegungen zur Tiefensuche erlauben es uns den ansonsten schwer zu führenden Beweis von Theorem III.3.6 mit Hilfe einer einfachen Induktion zu formulieren.

III.3.6. Theorem. *Algorithmus III.3 konstruiert für jedes wohlgeformte Spursystem $\mathcal{T}$ einen vollständigen Prozeßautomaten $\mathcal{A}$ zu $\mathcal{T}$.*

Beweis. Daß es sich bei $\mathcal{A}$ um einen Prozeßautmaten zu $\mathcal{T}$ handelt, folgt mit Hilfe einer Argumentation anlog zu der, die im Beweis zu Theorem III.2.13 verwendet wurde; es steht also nur die Vollständigkeit von $\mathcal{A}$ in Frage.

Wir zeigen: Ist $s \in S_{\mathcal{A}}$ und $a \in \mathrm{en}_{\mathcal{T}}(s)$ eine Aktion mit $F(a) \not\subseteq \mathrm{en}_{\mathcal{T}}(s) \Rightarrow F(a) \cap \mathrm{en}_{\mathcal{T}}(s) = \varnothing$, so gilt entweder

$$s \xRightarrow[\mathcal{A}]{a}, \tag{III.3.$\alpha$}$$

oder für alle Semiordnungen $x \in X_{\mathcal{A}}$, so daß $\delta_{\mathcal{A}}(s, x)$ definiert ist, gibt es eine Semiordnung $y \geqslant x$ und einen Zustand $\tilde{s} \in S_{\mathcal{A}}$ mit

$$s \xRightarrow[\mathcal{A}]{y} \tilde{s}\ \&\ s \xRightarrow[\mathcal{T}]{y\times a} \Rightarrow \tilde{s} \xRightarrow[\mathcal{A}]{a} \tag{III.3.$\beta$}$$

Weiterhin nehmen wir zunächst an, daß wenigstens eine weitere Aktion $b \in \mathrm{en}_{\mathcal{T}}(s)$ existiert, für die $F(b) \cap \mathrm{en}_{\mathcal{T}}(s) \neq \varnothing$) gilt.

Erinnern wir uns, daß $\mathcal{G}(\mathcal{A}) = \langle S_{\mathcal{A}}, R_{\mathcal{A}}, s_{\mathcal{A}}\rangle$ der Graph zu $\mathcal{A}$ ist. Der Beweis erfolgt durch eine Induktion über der Nummerierung $cnum(s)$ für Zustände $s \in S_{\mathcal{A}}$, wie sie durch Algorithmus III.4 berechnet wird; wir bezeichnen Baumkanten mit $R^{\mathrm{T}}_{\mathcal{A}}$, Vorwärtskanten mit $R^{\mathrm{F}}_{\mathcal{A}}$, Rückwärtskanten mit $R^{\mathrm{B}}_{\mathcal{A}}$ und Querkanten mit $R^{\mathrm{C}}_{\mathcal{A}}$.

Sei $s \in S_{\mathcal{A}}$, so daß $cnum(s) = 0$ gilt. Es gilt $s\ R_{\mathcal{A}}\ s' \Rightarrow s\ R^{\mathrm{B}}_{\mathcal{A}}\ s'$, da sonst mit Lemma III.3.5 $cnum(s') < 0$ folgen würde. Allerdings gilt $s\ R^{\mathrm{B}}_{\mathcal{A}}\ s' \Rightarrow s' \left(R^{\mathrm{T}-1}_{\mathcal{A}}\right)^*\ s$ (Zeile (9') in Algorithmus III.4). Es ist leicht einzusehen, daß $s\ T^{-1}\ s' \Leftrightarrow s'\ R^{\mathrm{T}}_{\mathcal{A}}\ s$

gilt; damit erhält die Variable *loop* in Zeile (11) den Wert true. In der Schleife (17) – (24) wird also ein Einzelschritt $\{a\}$ behandelt; es gilt $s \stackrel{a}{\underset{\mathcal{A}}{\Longrightarrow}}$.

Sei $cnum(s) > 0$. Nach Lemma III.3.5.(a) reicht es, die folgenden Fälle zu unterscheiden.

$s\ R^{\mathrm{T}}_{\mathcal{A}}\ s'$. Nach Lemma III.3.5.(b) gilt $cnum(s') < cnum(s)$ und damit die Induktionshypothese für $s'$. Bei der Konstruktion einer Semiordnung $x$ mit $\delta_{\mathcal{A}}(s,x) = s'$ werden zu $x$ nur Ereignisse $b$ mit $b\ I_\Sigma\ a$ hinzugefügt. Gilt also (III.3.$\alpha$) für $s'$, folgern wir

$$s \stackrel{x}{\underset{\mathcal{A}}{\Longrightarrow}} s' \ \&\ s \stackrel{x\times a}{\underset{\mathcal{T}}{\Longrightarrow}} \Rightarrow s' \stackrel{a}{\underset{\mathcal{A}}{\Longrightarrow}};$$

damit gilt (III.3.$\beta$) für $s$.

Gilt für $s'$ hingegen (III.3.$\beta$), so ersetzen wir in (III.3.$\beta$) $y$ durch $x \circ_\Sigma y$:

$$s \stackrel{x\circ_\Sigma y}{\underset{\mathcal{A}}{\Longrightarrow}} \tilde{s}\ \&\ s \stackrel{(x\circ_\Sigma y)\times a}{\underset{\mathcal{T}}{\Longrightarrow}} \Rightarrow \tilde{s} \stackrel{a}{\underset{\mathcal{A}}{\Longrightarrow}}$$

wiederum gilt nun (III.3.$\beta$) für $s$.

$s\ R^{\mathrm{F}}_{\mathcal{A}}\ s'$. Es gilt $cnum(s') < cnum(s)$ nach Lemma III.3.5.(d), die Argumentation entspricht dem Fall $s\ R^{\mathrm{T}}_{\mathcal{A}}\ s'$

$s\ R^{\mathrm{C}}_{\mathcal{A}}\ s'$. Wiederum ist $cnum(s') < cnum(s)$, diesmal mit Hilfe von Lemma III.3.5.(f). Wir schließen analog zum Fall $s\ R^{\mathrm{T}}_{\mathcal{A}}\ s'$.

$s\ R^{\mathrm{B}}_{\mathcal{A}}\ s'$. Nun gilt nach Lemma III.3.5.(e) $cnum(s') > cnum(s)$, die Induktionshypothese ist also nicht anwendbar. Allerdings haben wir den Fall $s\ R^{\mathrm{B}}_{\mathcal{A}}\ s'$ während unserer Überlegungen zum Induktionsanfang $cnum(s) = 0$ bereits diskutiert; diese Argumentation kann auch hier unverändert geführt werden.

Neben diesen Fällen, die möglicherweise in ihrer Betrachtung verschobene Aktionen $a$ behandeln, ist noch der Fall zu diskutieren, daß $F(a) \not\subseteq \mathrm{en}_{\mathcal{T}}(s) \Rightarrow F(a) \cap \mathrm{en}_{\mathcal{T}}(s) = \varnothing$ für alle $a \in \mathrm{en}_{\mathcal{T}}(s)$ gilt. In diesem Fall wird jedoch für jede dieser Aktionen $a$ ein Einzelschritt $\{a\}$ in der Schleife (17) – (24) betrachtet; wie zuvor folgt dann $s \stackrel{a}{\underset{\mathcal{A}}{\Longrightarrow}}$. □

III.3.7. Bemerkung. Wenn wir den Wert der Variablen $T$ unmittelbar vor der Termination von Algorithmus III.3 betrachten, so ist $\langle S_{\mathcal{A}}, T^{-1}, s_{\mathcal{A}}\rangle$ ein spannender Baum von $\mathcal{G}(\mathcal{A})$. Die Verwendung der Inversen zu $R^{\mathrm{T}}_{\mathcal{A}}$ erlaubt es, die Suche nach einem Knoten $s'$ mit $s' R^{\mathrm{T}*}_{\mathcal{A}} s$ leichter zu gestalten. □

III.3.8. Bemerkung. Leser, die mit der Theorie der *sturen Mengen* vertraut sind, werden die Ähnlichkeit der Kriterien zur Verschiebung der Betrachtung von Aktionen und der Lösung des sogenannten *ignoring problems* bei der Konstruktion stur reduzierter Transitionssysteme feststellen [**90**]. □

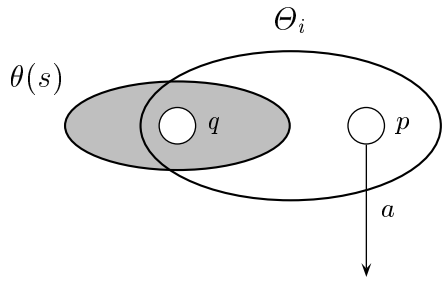



ABBILDUNG III.11. Komponenten zur Konfliktauflösung.

**Kleinere Konfliktrelationen — Komponenten.** In Bemerkung III.2.11 haben wir bereits angedeutet, daß für Netzsysteme die Wahl $F =_{\text{Def}} D_\Sigma - \text{id}_{A_\Sigma}$ für eine Vorwärtskonfliktrelation zu denkbar schlechten Ergebnissen des Erzeugungsalgorithmus III.1 für Prozeßautomaten führt, was seinen Speicherbedarf und damit seine Laufzeit anbelangt. Ähnliches gilt für die Wahl $B =_{\text{Def}} D_\Sigma - \text{id}_{A_\Sigma}$ einer Rückwärtskonfliktrelation. Es ist nicht zu erwarten, daß sich Algorithmus III.3 unter diesen Festlegungen wesentlich besser verhält. Mit Hilfe des in Abschnitt II.1 eingeführten Komponentenbegriffs jedoch können wir eine bessere Wahl treffen:

Erinnern wir uns, daß für jede Komponente $\Theta_i$ eines Spursystems $\mathcal{T}$ $|\Theta_i \cap \theta(s)| \leqslant 1$ gilt, wobei $\theta : S_{\mathcal{T}} \to \bigcup_i \Theta_i$ die zugehörige Positionszuweisung und $s \in S_{\mathcal{T}}$ ist. Ist $a \in \text{en}_{\mathcal{T}}(s)$, so gilt $\dot{\theta}(a) \subseteq \theta(s)$. Wenn es hingegen eine Position $p \in \Theta_i \cap \dot{\theta}(a)$ und eine weitere Position $q \in \Theta_i \cap \theta(s)$ mit $p \neq q$ gibt, so folgt $p \notin \theta(s)$ und damit $\dot{\theta}(a) \not\subseteq \theta(s)$, was wiederum $a \notin \text{en}_{\mathcal{T}}(s)$ impliziert. Abb. III.11 stellt die beschriebene Situation dar.

Wie hilft uns nun diese Überlegung bei unserer Suche nach einer kleineren Vorwärtskonfliktrelation? Wir suchen ein Kriterium, das uns für zwei Aktionen $a, b \in A_\Sigma$ mit $a \neq b$ und $a \; D_\Sigma \; b$ die Eigenschaft $\text{bp}_{a,b}(\mathcal{T}) = \varnothing$ zusichert. Wenn wir also $p, q \in \dot{\theta}(a) \cup \dot{\theta}(b)$ mit $p \neq q$ wählen, so daß eine Komponente $\Theta_i$ mit $p, q \in \Theta_i$ existiert, so folgt $\neg s \xrightarrow[\mathcal{T}]{a} \vee \neg s \xrightarrow[\mathcal{T}]{b}$ für alle $s \in S_{\mathcal{T}}$; da sowohl $p$ als auch $q$ in $\theta(s)$ vorhanden sein müssen, damit $a$ und $b$ simultan schaltfähig werden.

Mit Hilfe derselben Überlegung können wir auch die Zahl der Paare $\langle a, b\rangle \in B$ reduzieren, wenn $B$ eine Rückwärtskonfliktrelation ist. Es gilt nämlich $s \in \text{jp}_{a,b}(\mathcal{T}) \Rightarrow \theta^{\cdot}(a) \cup \theta^{\cdot}(b) \subseteq \theta(s)$. Finden wir also Positionen $p, q \in \theta^{\cdot}(a) \cup \theta^{\cdot}(b)$, so daß $p \neq q$ gilt und es weiterhin eine Komponente $\Theta_i$ mit $p, q \in \Theta_i$ gibt, so ist $\text{jp}_{a,b}(\mathcal{T}) = \varnothing$. Wir fassen diese Überlegungen in dem folgenden Theorem zusammen:

III.3.9. THEOREM. *Sei $\mathcal{T}$ ein Spursystem über $\Sigma$, für das es eine Positionszuweisung $\theta : \mathcal{T} \to \Theta$ gibt, so daß eine Familie $\boldsymbol{\Theta} = \{\Theta_i\}_{i\in J}$ von Komponenten mit $\Theta = \bigcup_{i\in J} \Theta_i$ existiert und $\mathcal{T}$ $\boldsymbol{\Theta}$-überdeckt ist. Seien $a, b \in A_\Sigma$ Aktionen mit $a \neq b$ und $a\ D_\Sigma\ b$. Wenn es eine Komponente $\Theta_i$ gibt, so daß*

$$|(\dot{\theta}(a) \cup \dot{\theta}(b)) \cap \Theta_i| > 1$$

*gilt, dann ist* $\mathrm{bp}_{a,b}(\mathcal{T}) = \varnothing$. *Gilt*

$$|(\theta^{\cdot}(a) \cup \theta^{\cdot}(b)) \cap \Theta_i| > 1,$$

*dann folgt* $\mathrm{jp}_{a,b}(\mathcal{T}) = \varnothing$.

**Kleinere Konfliktrelationen — Fallen.** Der Begriff der „Falle“, der seinen Ursprung in der Petri-Netz-Theorie hat, erlaubt es uns, eine weitere Technik zur Überprüfung der Nichterreichbarkeit von Zuständen $s \in \mathrm{bp}_{a,b}(\mathcal{T})$ bzw. $s \in \mathrm{jp}_{a,b}(\mathcal{T})$ zu formulieren. Eine *Falle* eines Petri-Netzes $\mathcal{N}$ ist eine nicht leere Menge von Plätzen $Q \subseteq P_\mathcal{N}$, so daß $Q^{\cdot} \subseteq {}^{\cdot}Q$ gilt, d. h. jede Transition, die eine Marke von einem der Plätze aus $Q$ entfernt, fügt eine Marke auf einem der Plätze in $Q$ hinzu.

Ist $\mathcal{T}$ ein Spursystem mit einer Positionszuweisung $\theta$, so nennen wir eine nicht leere Menge $Q \subseteq \Theta$ von Positionen eine *$\mathcal{T}$-Falle* bzw. kurz eine *Falle*, wenn $\theta^{\cdot}(Q) \subseteq \dot{\theta}(Q)$ gilt, wobei wir

$$a \in \dot{\theta}(Q) \Leftrightarrow_{\mathrm{Def}} Q \cap \theta^{\cdot}(a) \neq \varnothing \text{ und } a \in \theta^{\cdot}(Q) \Leftrightarrow_{\mathrm{Def}} Q \cap \dot{\theta}(a) \neq \varnothing$$

setzen. Offenbar gilt die Implikation

$$\theta(s) \cap Q \neq \varnothing \ \&\ s \overset{\sigma}{\underset{\mathcal{T}}{\Longrightarrow}} s' \Rightarrow \theta(s') \cap Q \neq \varnothing \qquad \text{(III.3.}\gamma\text{)}$$

für alle $\mathcal{T}$-Fallen $Q$, $s, s' \in S_\mathcal{T}$ und $\sigma \in A_\Sigma^*$; ein Beweis kann einfach durch Induktion über der Länge von $\sigma$ geführt werden.

Kombinieren wir nun die Begriffe der Komponente und der Falle: Sei $\boldsymbol{\Theta} = \{\Theta_i\}_{i\in J}$ eine Familie von Positionen, $a, b \in A_\Sigma$ Aktionen eines $\boldsymbol{\Theta}$-überdeckten Spursystems $\mathcal{T}$ über $\Sigma$, wobei $a \neq b$ und $a\ D_\Sigma\ b$ gelte. Die zugehörige Positionszuweisung sei $\theta : S_\mathcal{T} \to \Theta$ mit $\Theta = \bigcup_{i\in J} \Theta_i$. Dann ist

$$Q_{a,b} =_{\mathrm{Def}} \dot{\theta}(a) \cup \dot{\theta}(b)$$

die Menge derjenigen Positionen, die zur simultanen Konzessionierung von $a$ und $b$ in $\theta(s)$ für ein $s \in S_\mathcal{T}$ enthalten sein müssen; anders gesagt:

$$\forall s \in S_\mathcal{T} \left(s \in \mathrm{bp}_{a,b}(\mathcal{T}) \Rightarrow Q_{a,b} \subseteq \theta(s)\right).$$

Weiterhin definieren wir für eine Position $q \in \Theta$

$$\Theta_q =_{\mathrm{Def}} \bigcup_{i\in J} \{\Theta_i : q \in \Theta_i\}$$

**algorithm** *traps* **is**
  **input** $\tilde{Q}$, eine Positionsmenge für ein Spursystems $\mathcal{T}$;
  **output** $Q$, eine Falle, die in $\tilde{Q}$ enthalten ist;
  **local variables** $X$: **set of** $\tilde{Q}$;
**begin**
  $Q \leftarrow \tilde{Q}$;
  **loop**
    $X \leftarrow Q - \dot{\theta}\,(\theta^{\cdot}(Q) - \dot{\theta}(Q))$;
    **if** $Q = X$ **then exit else** $Q \leftarrow X$ **fi**
  **end**
**end** *traps*

ALGORITHMUS III.5. Berechnung maximaler Fallen.

als die Menge derjenigen Postitionen, die zu Komponenten gehören, zu der auch $q$ gehört. Schließlich sei

$$\tilde{Q}_{a,b} =_{\mathrm{Def}} \bigcup_{q \in Q_{a,b}} \Theta_q - Q_{a,b}.$$

Damit enthält $\tilde{Q}_{a,b}$ also gerade diejenigen Positionen, die nicht in $\theta(s)$ vorhanden sein können, wenn $a$ und $b$ simultan konzessioniert sind; mithin gilt $Q_{a,b} \cap \tilde{Q}_{a,b} = \varnothing$. Es folgt

$$\forall s \in S_{\mathcal{T}} \left( s \in \mathrm{bp}_{a,b}(\mathcal{T}) \Rightarrow \theta(s) \cap \tilde{Q}_{a,b} = \varnothing \right),$$

oder äquivalent

$$\forall s \in S_{\mathcal{T}} \left( \theta(s) \cap \tilde{Q}_{a,b} \neq \varnothing \Rightarrow s \notin \mathrm{bp}_{a,b}(\mathcal{T}) \right), \qquad \text{(III.3.}\delta\text{)}$$

Gibt es nun eine Falle $Q \subseteq \tilde{Q}_{a,b}$, so daß $\theta(s_{\mathcal{T}}) \cap Q \neq \varnothing$ ist, folgt $\mathrm{bp}_{a,b}(\mathcal{T}) = \varnothing$ mit (III.3.$\gamma$) und (III.3.$\delta$).

Setzen wir hingegen

$$Q_{a,b} =_{\mathrm{Def}} \theta^{\cdot}(a) \cup \theta^{\cdot}(b),$$

dann folgt aus der Existenz einer Falle $Q \subseteq \tilde{Q}_{a,b}$ mit $\theta(s_{\mathcal{T}}) \cap Q \neq \varnothing$, daß $\mathrm{jp}_{a,b}(\mathcal{T}) = \varnothing$ ist.

Die Vereinigung von Fallen ist wiederum eine Falle. Damit enthält jede Menge von Positionen eines Spursystems eine eindeutig bestimmte maximale Falle bzgl. der Mengeninklusion. Ein einfacher Algorithmus zur Berechnung solcher maximalen Fallen in Petri-Netzen ist in [**81**] angegeben, die offensichtliche Verallgemeinerung auf $\mathcal{T}$-Fallen ist als Algorithmus III.5 abgebildet. Man beachte, daß die

explizite Repräsentation des unterliegenden Spursystems zur Formulierung des Algorithmus nicht benötigt wird.

III.3.10. Bemerkung. Die Verwendung von Fallen in Verbindung mit anderen Methoden zum Nachweis der Nichterreichbarkeit von Zuständen findet ebenso Anwendung in der Verifikation nebenläufiger Systeme, die in diesem Kontext mit Hilfe sicherer Petri-Netze beschrieben werden [**7, 20, 56**]. □

KAPITEL IV

# Eine Logik für verteilte Systeme

Dieses Kapitel behandelt die temporale Logik DCTL sowie effektive Verfahren zum automatischen Nachweis, daß ein gegebenes verteiltes System eine durch eine DCTL-Formel beschriebene Eigenschaft aufweist. In Abschnitt IV.1 definieren wir die Syntax und die Semantik DCTL. Es zeigt sich, daß die Formulierung globaler Systemeigenschaften in der verteilten Logik DCTL nur schwer möglich ist. Wir stellen ein Verfahren zur Behandlung derartiger Eigenschaften vor. Abschnitt IV.2 erläutert anhand eines größeren Beispiels, wie die Spezifikation von Systemeigenschaften mit Hilfe von DCTL erfolgen kann. Abschnitt IV.3 beschreibt einen *Modelchecker* für DCTL.

Für dieses Kapitel legen wir fest, daß $\mathscr{D} = \langle \mathscr{T}_{\mathscr{D}}, \theta_{\mathscr{D}}, \boldsymbol{\Theta}_{\mathscr{D}} \rangle$ ein verteiltes System über $\Sigma$ und $J$ ist. Wir setzen $\boldsymbol{A}(\mathscr{D}) =_{\text{Def}} \{A_i\}_{i \in J}$. Der Einfachheit halber sei $J = \{1, 2, \ldots, N\}$.

## IV.1. DCTL

DCTL (D steht für *distributed*) ist eine agentenbasierte Version der klassischen temporalen Logik CTL (*computation tree logic*); zudem sind aktionsbasierte Schrittoperatoren enthalten (wir übersetzen *nexttime operator* mit Schrittoperator). Alle temporalen Operatoren beziehen sich auf einen spezifischen Agenten $i \in J$: Eine DCTL-Formel $\varphi$ erlangt nicht durch eine beliebige Ausführungsfolge eines verteilten Systems Gültigkeit, sondern durch Systementwicklungen, die lokal zu dem Agenten $i$ stattfinden. Dabei nennen wir eine Systementwicklung *lokal* zu $i$, wenn diese Entwicklung eine Folge von Schritten des Agenten $i$ sowie alle (und nur diese) Aktionen anderer Agenten enthält, die zum Stattfinden dieser Entwicklung notwendig sind. Bevor wir die Syntax und Semantik von DCTL formal definieren, wollen wir einige der DCTL-Operatoren informell beschreiben:

*Atomare Formeln.* Wie bei jeder propositionalen temporalen Logik wird die Existenz atomarer Aussagen angenommen. Der Wahrheitswert einer solchen Aussage $p$ wird als bekannt vorausgesetzt. In DCTL stehen atomare Aussagen in Verbindung mit einem Agenten $i \in J$: Atomare DCTL-Formeln haben die Form $p[i]$; die Gültigkeit einer solchen Formel wird dabei für einen Zustand bestimmt, der durch eine $i$-lokale Systementwicklung erreicht wird. Für andere Agenten mag $p$ nicht oder noch nicht gelten. Etwa kann es eine zur Entwicklung des Agenten $i$ nebenläufige Aktionsfolge eines Agenten $j$ geben, die nicht zu einem Systemzustand

führt, bei dem $p$ gilt. Oder $j$ partizipiert an der $i$-lokalen Entwicklung, die Gültigkeit von $p$ wird aber nicht direkt durch eine Aktion von $j$ hergestellt. In diesem Fall liegt das Eintreffen der Gültigkeit von $p$ in der lokalen Zukunft von $j$.

*Aktionsbasierte Schrittoperatoren.* Für jede Aktion $a$ des Agenten $i$ sind die Schrittoperatoren $\langle a \rangle_i \varphi$ und $[a]_i \varphi$ definiert. $\langle a \rangle_i \varphi$ bedeutet: Es gibt einen $a$-Schritt des Agenten $i$, nach dessen Stattfinden $\varphi$ gilt. $[a]_i \varphi$ besagt: Wenn $i$ einen $a$-Schritt vornimmt, so gilt $\varphi$ nach diesem Schritt. $\langle a \rangle_i$ heißt *unbedingter Schritt*, während wir $[a]_i$ als *bedingten Schritt* bezeichnen.

*Der aussagenlogische Teil.* Die üblichen aussagenlogischen Junktoren stehen in DCTL zur Verfügung: $\neg$ (Negation), $\vee$ (Disjunktion), $\oslash$ (exklusive Disjunktion), $\wedge$ (Konjunktion), $\supset$ (Implikation), $\equiv$ (Äquivalenz). Daneben gibt es die Konstanten **1** (Tautologie) und **0** (Kontradiktion).

*Aktionsunabhängige Schrittoperatoren.* $\boldsymbol{EX}_i\varphi$ gilt für eine gegebene (lokale) Systementwicklung, wenn es einen Schritt des Agenten $i$ gibt, nach dessen Ausführung $\varphi$ gilt. $\boldsymbol{AX}_i\varphi$ gilt, wenn $\varphi$ nach Ausführung aller möglichen $i$-Schritte gilt.

*„Von nun an" und „Letztendlich".* Wollten wir den Versuch unternehmen, für die DCTL-Operatoren das linguistische Gegenstück zu bestimmen, würden wir $\boldsymbol{G}_i$ mit „Von nun gilt ... (aus Sicht von $i$)" und $\boldsymbol{F}_i$ mit „Letztendlich wird ... (aus Sicht von $i$) gelten" übersetzen. Beide Operatoren tauchen in der existenz- und allquantorisierten Form auf; es ergeben sich also vier Kombinationen (wir lassen den Satzteil „(aus Sicht von $i$)" jeweils weg):

- $\boldsymbol{EG}_i\varphi$: „Von nun an gilt $\varphi$ für wenigstens eine $i$-lokale Entwicklung".
- $\boldsymbol{AG}_i\varphi$: „Von nun an gilt $\varphi$ für alle möglichen $i$-lokalen Entwicklungen".
- $\boldsymbol{EF}_i\varphi$: „Letztendlich wird bei wenigstens einer $i$-lokalen Entwicklung $\varphi$ gelten" (d. h. $\varphi$ ist aus Sicht von $i$ möglich) .
- $\boldsymbol{AF}_i\varphi$: „Letztendlich wird bei jeder $i$-lokalen Entwicklung $\varphi$ gelten" (d. h. $\varphi$ ist aus Sicht von $i$ notwendig).

*„Solange bis".* $\ldots_1 \boldsymbol{U}_i \ldots_2$ ist ein binärer Operator: „Es gilt $\ldots_1$, solange bis $\ldots_2$ gilt." Wieder gibt es die zwei Formen:

- $\boldsymbol{E}\,(\varphi\ \boldsymbol{U}_i\ \psi)$: „Es gibt eine $i$-lokale Entwicklung, bei der $\varphi$ gilt, solange bis $\psi$ wahr wird." Dabei muß $\psi$ irgendwann wahr werden.
- $\boldsymbol{A}\,(\varphi\ \boldsymbol{U}_i\ \psi)$: „Für alle $i$-lokalen Entwicklungen gilt $\varphi$, solange bis $\psi$ wahr wird." Wieder ist das Eintreffen der Gültigkeit von $\psi$ notwendig.

*Beschränkte Operatoren.* Ist $C \subseteq A_i$ eine Menge von Aktionen des Agenten $i$, so kann die Gültigkeit der Operatoren $\boldsymbol{X}_i$, $\boldsymbol{F}_i$ $\boldsymbol{G}_i$ und $\boldsymbol{U}_i$ auf $i$-lokale Systementwicklungen eingeschränkt werden, bei denen der Agent $i$ ausschließlich Aktionen

| | |
|---|---|
| Unäre Operatoren wie | |
| $\neg$, $\langle a\rangle_i$, $[a]_i$, $\boldsymbol{EX}_i$, $\boldsymbol{AF}_i$, $\boldsymbol{E}\,[C]\,\boldsymbol{G}_i$, ... | binden stärker als |
| $\wedge$ | bindet stärker als |
| $\vee, \veebar$ | binden stärker als |
| $\supset$ | bindet stärker als |
| $\equiv$ | |

Abbildung IV.1. Vorrangregeln für DCTL-Operatoren

aus $C$ verwendet. Syntaktisch bezeichnen wir solche auf $C$ beschränkten Operatoren, indem wir die Menge $C$ in eckige Klammern hinter dem jeweiligen Operator vorgestellten Quantor aufführen. Beispiele sind $\boldsymbol{E}\,[C]\,(\varphi\ \boldsymbol{U}_i\ \psi)$, $\boldsymbol{A}\,[C]\,\boldsymbol{G}_i\varphi$ oder $\boldsymbol{E}\,[C]\,\boldsymbol{F}_i\varphi$.

*Weitere Operatoren.* In Abschnitt IV.2 führen wir in Zusammenhang mit einem größeren Anwendungsbeispiel noch die Operatoren $\Longmapsto$ („bewirkt“) und $\boldsymbol{at}$ ein, die eine übersichtliche Darstellung von Eigenschaften erlauben, die für diesem Kontext spezifisch sind.

*Grundoperatoren und Ableitungen.* Wir verwenden die Operatoren $\langle a\rangle_i$, $\neg$, $\vee$, $\boldsymbol{E}\,(\cdot\ \boldsymbol{U}_i\ \cdot)$ und $\boldsymbol{A}\,(\cdot\ \boldsymbol{U}_i\ \cdot)$ als Grundoperatoren von DCTL, alle anderen Operatoren sind mit Hilfe dieser Grundoperatoren definierbar. Diese Wahl ist willkürlich: Zum Beispiel kann $[a]_i$ anstelle von $\langle a\rangle_i$ oder $\boldsymbol{EG}_i$ anstelle von $\boldsymbol{A}\,(\cdot\ \boldsymbol{U}_i\ \cdot)$ verwendet werden.

**Syntax von DCTL.** DCTL-Formeln sind unter Bezugnahme auf eine Menge atomarer Aussagen $P$ und eine Familie von Aktionsalphabeten $\boldsymbol{A} = \{A_i\}_{i\in J}$ definiert; wenn wir die Semantik von DCTL beschreiben, werden wir $\boldsymbol{A} = \boldsymbol{A}(\mathcal{D})$ annehmen.

IV.1.1. Definition (Syntax von DCTL). Sei $P$ eine Menge *atomarer Aussagen* und $\boldsymbol{A} = \{A_i\}_{i\in J}$ eine Familie endlicher nicht leerer Mengen für eine endliche nicht leere Indexmenge $J$. Die Menge $\Phi(P, \boldsymbol{A})$ der DCTL-Formeln ist induktiv wie folgt definiert:

(a) Für $p \in P$ und $i \in J$ ist $p[i] \in \Phi(P, \boldsymbol{A})$,
(b) sind $\varphi, \psi \in \Phi(P, \boldsymbol{A})$, so sind auch $\varphi \vee \psi$ und und $\neg\varphi$ in $\Phi(P, \boldsymbol{A})$ enthalten,
(c) für $i \in J$, $a \in A_i$ und $\varphi, \psi \in \Phi(P, \boldsymbol{A})$ sind $\langle a\rangle_i\,\varphi$, $\boldsymbol{E}\,(\varphi\ \boldsymbol{U}_i\ \psi)$ und $\boldsymbol{A}\,(\varphi\ \boldsymbol{U}_i\ \psi)$ in $\Phi(P, \boldsymbol{A})$ enthalten.

□

Wir nehmen uns die Freiheit, zusätzliche Klammerungen $(,)$, $\langle,\rangle$, $[,]$ oder $\{,\}$ in DCTL-Formeln zu verwenden, wann immer dies der Übersichtlichkeit dient. Mit Hilfe der in Abbildung IV.1 zusammengestellten Regeln können Klammern

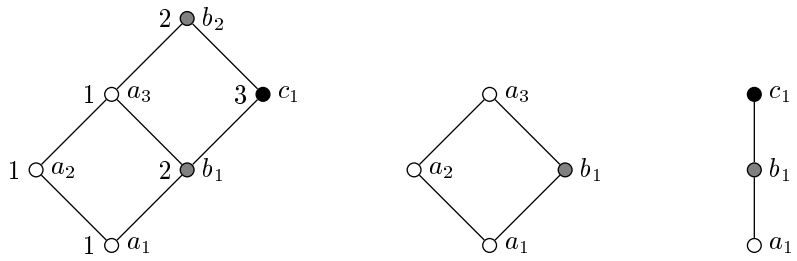



ABBILDUNG IV.2. Illustration zum Begriff der $i$-Sicht.

eliminiert werden (einige der genannten Operatoren werden erst im folgenden eingeführt). Wir verwenden die üblichen Schreibweisen wie

$$\bigvee_{i=1}^{k} \varphi_i,$$

um DCTL-Formeln der Form $\varphi_1 \vee \varphi_2 \vee \cdots \vee \varphi_k$ zu bezeichnen.

**Agentensichten.** Die Semantik von DCTL-Formeln ist relativ zu $i$-lokalen Systementwicklungen definiert. Wir formalisieren nun den Begriff der $i$-lokalen Entwicklung.

IV.1.2. DEFINITION (Agentensicht). Sei $x \in \mathbf{CSO}(\Sigma)$. Die *Sicht* des Agenten $i \in J$ auf $x$ ist

$$\downarrow^i x \;=_{\mathrm{Def}}\; x\left[\leqslant_x^{-1}\left(\lambda_x^{-1}(A_i) \cap E_x\right)\right]$$

Ist $\boldsymbol{x} \in \mathbf{CSW}(\Sigma)$, so setzen wir $\downarrow^i \boldsymbol{x} =_{\mathrm{Def}} [\downarrow^i x]$, die Operation $\downarrow^i(\cdot) : \mathbf{CSW}(\Sigma) \to \mathbf{CSW}(\Sigma)$ ist dann offensichtlich wohldefiniert. Ist $K \subseteq J$ und $\boldsymbol{x} \in \mathbf{CSW}(\Sigma)$, so ist

$$\downarrow^K \boldsymbol{x} \;=_{\mathrm{Def}}\; \bigvee_{k \in K} \downarrow^k \boldsymbol{x}.$$

Schließlich bezeichnen wir die Menge der *$i$-lokalen Konfigurationen* in $\mathscr{D}$ mit

$$\mathbf{CNF}_i(\mathscr{D}) =_{\mathrm{Def}} \left\{\downarrow^i \boldsymbol{x} : \boldsymbol{x} \in \mathbf{LSSL}(\mathscr{D})\right\}.$$

□

IV.1.3. BEISPIEL. Abb. IV.2 zeigt ein Semiwort $\boldsymbol{x}$. Den Aktionen $a_1$, $a_2$, $a_3$, $b_1$, $b_2$ und $c_1$ sind Agenten 1, 2 und 3 in der gezeigten Weise zugeordnet. Weiterhin sind die 1-Sicht sowie die 3-Sicht auf $\boldsymbol{x}$ abgebildet; die 2-Sicht auf $\boldsymbol{x}$ ist identisch zu $\boldsymbol{x}$. □

IV.1.4. LEMMA. *Für jedes Semiwort* $\boldsymbol{x} \in \mathbf{LSSL}(\mathscr{D})$ *und jedes* $i \in J$ *gilt:*

(a) $\lambda_x^{-1}(A_i) \cap E_x$ *ist eine Kette in* $x$.
(b) *Ist* $\downarrow^i \boldsymbol{x} \neq \boldsymbol{\epsilon}$, *dann gibt es ein eindeutig bestimmtes* $e \in E_x$, *so daß* $\downarrow^i \boldsymbol{x} = [x\,[\leqslant_x^{-1}(e)]]$ *ist, nämlich das maximale Ereignis* $e$ *mit* $\lambda_x(e) \in A_i$.
(c) *Die Menge* $\{\downarrow^i \boldsymbol{y} : \boldsymbol{y} \leqslant \boldsymbol{x}\}$ *ist linear geordnet unter* $\leqslant$.

BEWEIS. (a) folgt mit Theorem II.3.3 und dem Umstand, daß $\boldsymbol{x} \in \mathbf{CSW}(\Sigma)$ ist.

(b) Da nach (a) $\lambda_a^{-1}(A_i) \cap E_x$ eine Kette ist, enthält diese Menge ein eindeutiges maximales Ereignis $e$. Die Behauptung folgt nun mit $e' <_x e \Leftrightarrow \leqslant_x^{-1}(e') \subseteq \leqslant_x^{-1}(e)$.

(c) Nehmen wir Semiwörter $\boldsymbol{y}, \boldsymbol{z}$ mit $\boldsymbol{y} \leqslant \boldsymbol{x}$ und $\boldsymbol{z} \leqslant \boldsymbol{x}$ an. Offenbar gilt $\downarrow^i \boldsymbol{y} \leqslant \boldsymbol{y}$ und $\downarrow^i \boldsymbol{z} \leqslant \boldsymbol{z}$. Sei $\{e_1\} = \max(\downarrow^i y)$ und $\{e_2\} = \max(\downarrow^i z)$. Mit (a) folgt $H^x_{\downarrow^i y}(e_1) = e_1'$ $\mathrm{li}_x$ $e_2' = H^x_{\downarrow^i z}(e_2)$, o. B. d. A. nehmen wir $e_1' <_x e_2'$ an. Man vergegenwärtige sich für die folgende Rechnung die Ausführungen über kanonische Semiordnungen aus Abschnitt I.2:

$$\begin{aligned}
e_1' <_x e_2' &\Rightarrow <_x^{-1}(e_1') \subseteq <_x^{-1}(2_1') \\
&\Rightarrow \gamma_x\left(<_x^{-1}(e_1')\right) \subseteq \gamma_x\left(<_x^{-1}(e_2')\right) && \text{(Theorem I.2.29.(a))} \\
&\Rightarrow E_{\gamma(\downarrow^i y)} \subset E_{\gamma(\downarrow^i z)} && \text{(Def. } \gamma(\downarrow^i y),\ \gamma(\downarrow^i z)) \\
&\Rightarrow \downarrow^i y < \downarrow^i z. && \text{(Theorem I.2.29.(e))}
\end{aligned}$$

□

Es ist interessant, die Auswirkungen von Operationen auf Agentenmengen auf Sichten zu beobachten:

IV.1.5. LEMMA. *Sei* $\boldsymbol{x} \in \mathbf{CSW}(\Sigma)$. *Für alle* $L, K \subseteq J$ *gilt:*

(a) $\downarrow^{K \cap L} \boldsymbol{x} = \downarrow^K \boldsymbol{x} \wedge \downarrow^L \boldsymbol{x}$,
(b) $\downarrow^{K \cup L} \boldsymbol{x} = \downarrow^K \boldsymbol{x} \vee \downarrow^L \boldsymbol{x}$,
(c) $K \subseteq L \Rightarrow \downarrow^K \boldsymbol{x} \leqslant \downarrow^L \boldsymbol{x}$,
(d) $\downarrow^{\varnothing} \boldsymbol{x} = \boldsymbol{\epsilon}$,
(e) $\downarrow^J \boldsymbol{x} = \boldsymbol{x}$.

BEWEIS. Direkte Anwendung der entsprechenden Definitionen, wobei die notwendigen Berechnungen günstig auf Ereignismengen kanonischer Semiordnungen durchgeführt werden können. □

**Semantik von DCTL.** DCTL liegt ein *diskretes Zeitmodell* zugrunde: Ein Fortschritt in einem verteilten System (einem Modell für DCTL) findet in diskreten Schritten statt. Solche Schritte werden durch Systemaktionen $a \in A_\Sigma$ bestimmt, finden also (da $a \in A_i$ für wenigstens einen Agenten $i \in J$ ist) relativ zu einem der Agenten $i$ statt. Damit überführen Schritte $i$-lokale Konfigurationen in andere $i$-lokale Konfigurationen, eine Schrittrelation ist also auf der Menge $\mathbf{CNF}_i(\mathscr{D})$ zu definieren.

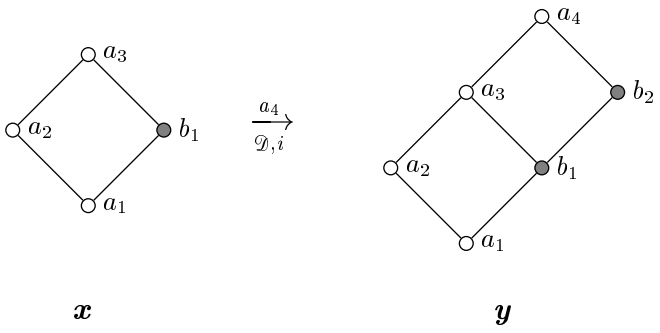



ABBILDUNG IV.3. Illustration der $i$-Schrittrelation.

IV.1.6. DEFINITION. Sei $i \in J$ und $a \in A_i$. Wir definieren eine *Schrittrelation, lokale* $\xrightarrow[\mathscr{D},i]{a} \subseteq \mathbf{CNF}_i(\mathscr{D}) \times \mathbf{CNF}_i(\mathscr{D})$ für lokale Konfigurationen:

$$\boldsymbol{x} \xrightarrow[\mathscr{D},i]{a} \boldsymbol{y} \quad \Leftrightarrow_{\text{Def}} \quad \boldsymbol{x} < \boldsymbol{y} \;\&\; \{a\} = \lambda_y(\max(y))$$
$$\&\; \forall \boldsymbol{z} \in \mathbf{LSSL}(\mathscr{D})\,(\boldsymbol{x} \leqslant \boldsymbol{z} \leqslant \boldsymbol{y} \Rightarrow \boldsymbol{z} \notin \mathbf{CNF}_i(\mathscr{D}))$$

Weiterhin setzen wir

$$\xrightarrow[\mathscr{D}]{i} =_{\text{Def}} \bigcup_{a \in A_i} \xrightarrow[\mathscr{D},i]{a}.$$

□

IV.1.7. BEISPIEL. Abb. IV.3 illustriert den Begriff der $i$-Schrittrelation. Abgebildet sind $i$-lokale Konfigurationen $\boldsymbol{x}$ und $\boldsymbol{y}$, die in einem verteilten Systems $\mathscr{D}$ als schaltfähig angenommen werden. Neben dem Agenten $i$, dem die Aktionen $a_1$, $a_2$, $a_3$ und $a_4$ zugeordnet sind, kommt noch ein weiterer, von $i$ verschiedener Agent vor, dem die Aktionen $b_1$ und $b_2$ zugeordnet sind (grau unterlegt). Man beachte, daß ein $i$-Schritt durchaus einen Schritt oder mehrere Schritte anderer Agenten beinhalten kann. □

Zur Definition der Semantik von Varianten von CTL wird üblicherweise der Begriff der unendlichen Aktions- oder Zustandsfolge verwendet. Systeme, die Verklemmungen enthalten (also solche Systeme, in denen endliche, nicht weiter fortsetzbare Aktionsfolgen stattfinden können), werden durch das Hinzufügen von „Schlingen“ verklemmungsfrei gemacht: Es wird eine neue Systemaktion $*$ zu $\Sigma$ hinzugefügt und $\delta_{\mathscr{D}}(s,*) = s$ für jeden terminalen Zustand $s \in S_{\mathscr{D}}$ gesetzt. Das impliziert jedoch, daß (zumindest in nicht aktionsbasierten CTL-Varianten) nicht ohne weiteres Aussagen über Verklemmungen gemacht werden können.

Als Alternative kann mit dem Begriff des maximalen Weges gearbeitet werden.

IV.1.8. Definition. Wir nennen eine endliche oder unendliche Folge $\boldsymbol{\xi} = \boldsymbol{x}_0\boldsymbol{x}_1\ldots\boldsymbol{x}_j\ldots$ von Semiwörtern $\boldsymbol{x}_j \in \mathbf{CNF}_i(\mathscr{D})$ einen *$i$-Weg* oder kurz *Weg* beginnend bei $\boldsymbol{x}_0$ durch $\mathbf{LSSL}(\mathscr{D})$, wenn $\boldsymbol{x}_j \xrightarrow[\mathscr{D}]{i} \boldsymbol{x}_{j+1}$ für $j \geqslant 0$ gilt.[1] $\boldsymbol{\xi}$ heißt *maximal*, wenn $\boldsymbol{\xi}$ entweder unendlich oder aber endlich und der Form $\xi = \boldsymbol{x}_0\boldsymbol{x}_1\ldots\boldsymbol{x}_{n-1}$ ist und zusätzlich $\neg\boldsymbol{x}_{n-1} \xrightarrow[\mathscr{D},i]{a} \boldsymbol{y}$ für alle $a \in A_i$ und für alle $\boldsymbol{y} \in \mathbf{CNF}_i(\mathscr{D})$ gilt. Mit $\mathbf{P}_{\mathscr{D},i}(\boldsymbol{x})$ bezeichnen wir die Menge aller bei $\boldsymbol{x}$ beginnenden maximalen Wege.

Für Wege $\boldsymbol{\xi} \in \mathbf{P}_{\mathscr{D},i}(\boldsymbol{x})$ verwenden wir die folgenden Schreibweisen:

(a) Ist $\boldsymbol{\xi} = \boldsymbol{x}_0\boldsymbol{x}_1\ldots\boldsymbol{x}_{n-1}$ endlich, so bezeichnen wir mit $|\boldsymbol{\xi}| =_{\text{Def}} n$ die *Länge* von $\boldsymbol{\xi}$. Ist $\boldsymbol{\xi}$ unendlich, so setzen wir $|\boldsymbol{\xi}| =_{\text{Def}} \infty$. Um die folgenden Definitionen einfach zu halten, nehmen hierbei $n < \infty$ für alle $n \in \mathbb{N}$ an.

(b) Ist $j < |\boldsymbol{\xi}|$, so ist $\boldsymbol{\xi}(j) =_{\text{Def}} \boldsymbol{x}_j$.

□

Wir beschreiben nun die Semantik von DCTL. Wie üblich verwenden wir die Schreibweise $\mathscr{D}, \boldsymbol{x} \vDash \varphi$, um auszudrücken, daß eine Formel $\varphi$ bei $\boldsymbol{x} \in \mathbf{CNF}_i(\mathscr{D})$ in $\mathscr{D}$ gilt; ist dies der Fall, so nennen wir das Paar $\langle\mathscr{D}, \boldsymbol{x}\rangle$ ein *Modell* von $\varphi$. Dabei steht $\boldsymbol{x}$ stellvertretend für den Zustand $s = \delta_{\mathscr{D}}(s_{\mathscr{D}}, x)$; wie wir sehen werden (Lemma IV.1.12), hängt die Gültigkeit von $\varphi$ lediglich von $s$, nicht jedoch von $\boldsymbol{x}$ ab.

Eine DCTL-Formel $\varphi$ gilt in einem verteilten System $\mathscr{D}$, wenn $\langle\mathscr{D}, \boldsymbol{\epsilon}\rangle$ ein Modell von $\varphi$ ist; wir sagen dann, $\mathscr{D}$ ist ein Modell für $\varphi$ und schreiben $\mathscr{D} \vDash \varphi$.

IV.1.9. Definition (Semantik von DCTL). Sei $P$ eine Menge atomarer Aussagen. Es sei eine Familie von *Wertzuweisungen* $\nu = \{\nu_i : \mathbf{CNF}_i(\mathscr{D}) \to \mathscr{P}(P)\}_{i\in J}$ definiert, so daß für alle $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{CNF}_i(\mathscr{D})$ gilt:

$$\delta_{\mathscr{D}}(s_{\mathscr{D}}, x) = \delta_{\mathscr{D}}(s_{\mathscr{D}}, y) \Rightarrow \nu_i(\boldsymbol{x}) = \nu_i(\boldsymbol{y})$$

Sei $\varphi \in \Phi(P, \boldsymbol{A}(\mathscr{D}))$ und $\boldsymbol{x} \in \mathbf{CNF}_i(\mathscr{D})$. Die Relation $\vDash$ wird induktiv über dem Aufbau von DCTL-Formeln definiert:

(a) Für alle $p \in P$: $\mathscr{D}, \boldsymbol{x} \vDash p[i]$ gdw. $p \in \nu_i(\boldsymbol{x})$ gilt

(b) $\mathscr{D}, \boldsymbol{x} \vDash \neg\varphi$ gdw. $\mathscr{D}, \boldsymbol{x} \vDash \varphi$ nicht gilt.

(c) $\mathscr{D}, \boldsymbol{x} \vDash \varphi \vee \psi$ gdw. $\mathscr{D}, \boldsymbol{x} \vDash \varphi$ oder $\mathscr{D}, \boldsymbol{x} \vDash \psi$ gilt.

(d) $\mathscr{D}, \boldsymbol{x} \vDash \langle a\rangle_i\,\varphi$ gdw. es ein $\boldsymbol{y} \in \mathbf{CNF}_i(\mathscr{D})$ mit $\boldsymbol{x} \xrightarrow[\mathscr{D},i]{a} \boldsymbol{y}$ und $\mathscr{D}, \boldsymbol{y} \vDash \varphi$ gibt.

(e) $\mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{E}\,(\varphi\ \boldsymbol{U}_i\ \psi)$ gdw. es ein $\boldsymbol{\xi} \in \mathbf{P}_{\mathscr{D},i}(\boldsymbol{x})$ und ein $j < |\boldsymbol{\xi}|$ mit $\mathscr{D}, \boldsymbol{\xi}(j) \vDash \psi$ gibt und weiterhin $\mathscr{D}, \boldsymbol{\xi}(k) \vDash \varphi$ für $0 \leqslant k < j$ gilt.

(f) $\mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{A}\,(\varphi\ \boldsymbol{U}_i\ \psi)$ gdw. für alle Wege $\boldsymbol{\xi} \in \mathbf{P}_{\mathscr{D},i}(\boldsymbol{x})$ ein $j < |\boldsymbol{\xi}|$ mit $\mathscr{D}, \boldsymbol{\xi}(j) \vDash \psi$ existiert und weiterhin $\mathscr{D}, \boldsymbol{\xi}(k) \vDash \varphi$ für $0 \leqslant k < j$ gilt.

Schließlich schreiben wir $\mathscr{D} \vDash \varphi$, wenn $\mathscr{D}, \boldsymbol{\epsilon} \vDash \varphi$ gilt (man beachte $\boldsymbol{\epsilon} \in \mathbf{CNF}_i(\mathscr{D})$ für alle $i \in J$). □

[1] Zur Präzisierung des Begriffs der endlichen bzw. unendlichen Folge und der Zählweise: Eine *endliche Folge* über $\mathbf{CNF}_i(\mathscr{D})$ ist eine Abbildung $[n] \to \mathbf{CNF}_i(\mathscr{D})$, eine *unendliche Folge* ist eine Abbildung $\mathbb{N} \to \mathbf{CNF}_i(\mathscr{D})$.

Bei unserer Definition der Gültigkeitsrelation ist ihre Einschränkung auf $i$-lokale Konfigurationen $\boldsymbol{x}$ zu beachten. Wir wollen diesen Punkt mit Hilfe der folgenden Begriffe verdeutlichen: Bestimmen wir zunächst, auf welche Agentenmenge sich eine DCTL-Formel bezieht: Die Abbildung $\mathrm{loc} : \Phi(P, \boldsymbol{A}(\mathcal{D})) \to \mathcal{P}(J)$ ist induktiv über dem Aufbau von DCTL-Formeln definiert:

$$\mathrm{loc}(p[i]) =_{\mathrm{Def}} \{i\}, \mathrm{loc}(\neg\varphi) =_{\mathrm{Def}} \mathrm{loc}(\varphi), \mathrm{loc}(\varphi \vee \psi) =_{\mathrm{Def}} \mathrm{loc}(\varphi) \cup \mathrm{loc}(\psi),$$
$$\mathrm{loc}(\langle a \rangle_i \varphi) =_{\mathrm{Def}} \{i\}, \mathrm{loc}(\boldsymbol{E}(\varphi \; \boldsymbol{U}_i \; \psi)) =_{\mathrm{Def}} \{i\}, \mathrm{loc}(\boldsymbol{A}(\varphi \; \boldsymbol{U}_i \; \psi)) =_{\mathrm{Def}} \{i\}.$$

Definieren wir die Menge der *$i$-getypten* Formeln als

$$\Phi_i(P, \boldsymbol{A}(\mathcal{D})) =_{\mathrm{Def}} \{\varphi \in \Phi(P, \boldsymbol{A}(\mathcal{D})) : \mathrm{loc}(\varphi) = \{i\}\}$$

Es ist nun leicht zu sehen, daß für jede DCTL-Formel $\varphi$ eine Menge $K \subseteq J$ und eine Familie von Formeln $\{\varphi_i\}_{i \in K}$ mit $\varphi_i \in \Phi_i(P, \boldsymbol{A}(\mathcal{D}))$ für alle $i \in K$ existiert, so daß $\varphi = \bigvee_{i \in K} \varphi_i$ ist. Wenn wir nun Def. IV.1.9 noch einmal vorsichtig lesen, stellen wir fest:

$$\mathcal{D}, \boldsymbol{x} \models \varphi \;\&\; \varphi \in \Phi_i(P, \boldsymbol{A}(\mathcal{D})) \Rightarrow \boldsymbol{x} \in \mathbf{CNF}_i(\mathcal{D})$$

bzw. allgemeiner

$$\mathcal{D}, \boldsymbol{x} \models \varphi \Rightarrow \boldsymbol{x} \in \bigcup_{i \in \mathrm{loc}(\varphi)} \mathbf{CNF}_i(\mathcal{D}).$$

Bei der Auswertung einer Formel $\varphi$ bei einer lokalen Konfiguration $\boldsymbol{x}$ wird diese als lokal zu einem der Elemente aus $\mathrm{loc}(\varphi)$ behandelt. Wenn $\psi$ also eine $j$-getypte Teilformel einer $i$-getypten Formel $\varphi$ mit $\mathcal{D}, \boldsymbol{x} \models \varphi$ ist, dann unterhält der Agent $i$ eine direkte Kausalbeziehung mit dem Agenten $j$, d. h. es gibt eine Aktion $a \in A_i \cap A_j$, so daß ein $i$-Weg zu einem mit $a$ beschrifteten Ereignis in $\boldsymbol{x}$ führt, das wiederum diejenigen $j$-Wege einleitet, die zur Gültigkeit von $\psi$ führen.

IV.1.10. Beispiel. Abb. IV.4 illustriert diese „Weitergabe der Kontrolle“ von einem Agenten an einen anderen für die Formel $\boldsymbol{E}(\mathbf{1} \; \boldsymbol{U}_i \; \boldsymbol{E}(\mathbf{1} \; \boldsymbol{U}_j \; \langle a \rangle_k \; \mathbf{1}))$.[2] Es gibt zunächst einen $i$-Weg, der ausgehend vom Initialzustand $s_{\mathcal{D}}$ von $\mathcal{D}$ zu einem Ereignis führt, das mit einer Aktion $b \in A_i \cap A_j$ beschriftet ist. An dieser Stelle werden zur Auswertung der Teilformel $\boldsymbol{EF}_j \langle a \rangle_k \mathbf{1}$ $j$-Wege betrachtet, die wiederum zu einem mit $c \in A_j \cap A_k$ beschrifteten Ereignis führen. An dieser Stelle übernimmt der Agent $k$ die Kontrolle, indem er die Systementwicklung mit der Aktion $a$ fortsetzt. □

Dieser subtile Punkt in der Definition der $\models$-Relation hätte explizit gemacht werden können, indem stattdessen eine Familie von Relation $\{\models_i\}_{i \in J}$ definiert worden wäre, wobei die Selektion einer der Relation $\models_i$ aufgrund des Typs ihres Formelarguments stattgefunden hätte. Allerdings hätte diese Vorgehensweise die Kompliziertheit von Def. IV.1.9 stark erhöht; es ist unwahrscheinlich, daß dadurch die Semantik von DCTL verständlicher geworden wäre.

[2] Diese Formel ist äquivalent zu $\boldsymbol{EF}_i \boldsymbol{EF}_j \langle a \rangle_k \mathbf{1}$, s. u.

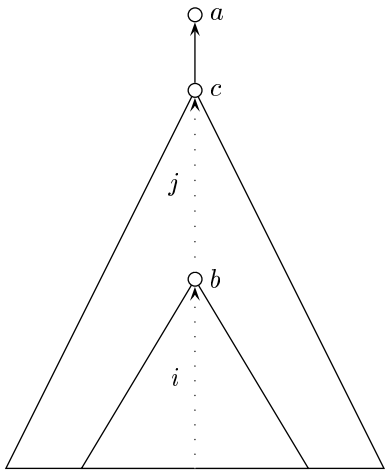



ABBILDUNG IV.4. Illustration zur Interpretation von Teilformeln.

---

IV.1.11. BEMERKUNG. Die in [**71, 85**] vorgestelle Logik TrPTL unterscheidet sich (auch) in diesem Punkt von DCTL: In TrPTL kann die „Weitergabe der Kontrolle“ von $i$ nach $j$ auch indirekt erfolgen, indem die Gültigkeit einer $j$-getypten Teilformel bei einer $i$-lokalen Konfiguration $\boldsymbol{x}$ durch die größte $j$-lokale Konfiguration $\boldsymbol{y}$ bestimmt ist, die ein Präfix von $\boldsymbol{x}$ ist. Das impliziert, daß in TrPTL Aussagen formulierbar sind, die sich auf die Vergangenheit einer Systementwicklung beziehen. Insbesondere gilt das Analogon von Lemma IV.1.12 für TrPTL nicht.

Dies führt zu einem relativ komplizierten Modelchecking-Algorithmus. In [**71, 85**] wurde deshalb die Klasse der erlaubten Formeln syntaktisch auf sog. *verbundene* Formeln eingeschränkt. Unsere Vorgehensweise entspricht dem in [**40**] vorgestellen Ansatz, bei dem Verbundenheit auf semantischer Ebene sichergestellt wird. □

IV.1.12. LEMMA. *Seien* $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{CNF}_i(\mathscr{D})$ *$i$-lokale Konfigurationen, so daß* $\delta_{\mathscr{D}}(s_{\mathscr{D}}, x) = \delta_{\mathscr{D}}(s_{\mathscr{D}}, y)$ *gilt. Ist* $\varphi \in \Phi(P, \boldsymbol{A}(\mathscr{D}))$*, so gilt* $\mathscr{D}, \boldsymbol{x} \vDash \varphi \Leftrightarrow \mathscr{D}, \boldsymbol{y} \vDash \varphi$

BEWEIS. Eine einfache Induktion über der Struktur von DCTL-Formeln $\varphi \in \Phi(P, \boldsymbol{A}(\mathscr{D}))$. □

IV.1.13. DEFINITION (Äquivalente Formeln). Wir nennen zwei DCTL-Formeln $\varphi$ und $\psi$ *logisch äquivalent*, wenn

$$\mathscr{D}, \boldsymbol{x} \vDash \varphi \Leftrightarrow \mathscr{D}, \boldsymbol{x} \vDash \psi$$

für alle verteilten Systeme $\mathscr{D}$ über $\Sigma$ und einer Agentenmenge $J$, für alle $i \in J$ und für alle $\boldsymbol{x} \in \mathbf{CNF}_i(\mathscr{D})$ gilt. Sind $\varphi$ und $\psi$ äquivalent, so schreiben wir $\varphi \Leftrightarrow \psi$. □

---

**Abgeleitete Operatoren I.** Wie in CTL können auch in DCTL eine Reihe abgeleiteter Operatoren definiert werden. Wir beginnen (der Vollständigkeit halber) mit den aussagenlogischen Ableitungen:

IV.1.14. DEFINITION (Aussagenlogische Konstanten und Junktoren). Die folgenden abgeleiteten Operatoren seien definiert (man beachte die Regeln zur Bindungsstärke der DCTL-Operatoren):

$$\begin{aligned}
\mathbf{1} &=_{\text{Def}} \bigvee_{i\in J}(p[i] \vee \neg p[i]), && (p \in P \text{ beliebig})\\
\mathbf{0} &=_{\text{Def}} \neg\mathbf{1},\\
\varphi \wedge \psi &=_{\text{Def}} \neg(\neg\varphi \vee \neg\psi),\\
\varphi \mathbin{⊛} \psi &=_{\text{Def}} \varphi \wedge \neg\psi \vee \neg\varphi \wedge \psi,\\
\varphi \supset \psi &=_{\text{Def}} \neg\varphi \vee \psi,\\
\varphi \equiv \psi &=_{\text{Def}} (\varphi \supset \psi) \wedge (\psi \supset \varphi).
\end{aligned}$$

□

Der duale Operator zu $\langle a\rangle_i\,\varphi$ ist $[a]_i\,\varphi$ mit der Bedeutung: Für alle möglichen $a$-Schritte des Agenten $i$ gilt $\varphi$ nach Ausführung dieses Schritts. Man beachte hierbei aber, daß wir deterministische Spursysteme betrachten, d. h. in jedem Zustand eines Spursystems kann der Agent $i$ höchstens einen $a$-Schritt vornehmen.

IV.1.15. DEFINITION (Bedingter Schritt). Der Operator $[a]_i$ (*bedingter Schritt*) ist dual zu $\langle a\rangle_i$:

$$[a]_i\,\varphi =_{\text{Def}} \neg\,\langle a\rangle_i\,\neg\varphi.$$

Eine direkte Definition der Semantik dieses Operators ist:
$\mathscr{D}, \boldsymbol{x} \vDash [a]_i\,\varphi$ gdw. $\boldsymbol{x} \in \mathbf{CNF}_i(\mathscr{D})$ ist und für alle $\boldsymbol{y} \in \mathbf{CNF}_i(\mathscr{D})$ die Implikation $\boldsymbol{x} \xrightarrow[\mathscr{D},i]{a} \boldsymbol{y} \Rightarrow \mathscr{D}, \boldsymbol{y} \vDash \varphi$ gilt. □

IV.1.16. DEFINITION ($i$-lokale CTL-Operatoren). Die folgenden Operatoren sind $i$-lokale Versionen der gebräuchlichen CTL-Operatoren. Wir geben jeweils die Ableitung der Operatoren und eine direkte Definition ihrer Semantik an; dabei sei jeweils $\boldsymbol{x} \in \mathbf{CNF}_i(\mathscr{D})$

(a) $\boldsymbol{EX}_i\varphi =_{\text{Def}} \bigvee_{a\in A_i} \langle a\rangle_i\,\varphi$;
$\mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{EX}_i\varphi$ gdw. es ein $\boldsymbol{y} \in \mathbf{CNF}_i(\mathscr{D})$ mit $\boldsymbol{x} \xrightarrow[\mathscr{D}]{i} \boldsymbol{y}$ und $\mathscr{D}, \boldsymbol{y} \vDash \varphi$ gibt.

(b) $\boldsymbol{AX}_i\varphi =_{\text{Def}} \bigwedge_{a\in A_i} [a]_i\,\varphi$ bzw. $\boldsymbol{AX}_i\varphi =_{\text{Def}} \neg\boldsymbol{EX}_i\neg\varphi$;
$\mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{AX}_i\varphi$ gdw. für alle $\boldsymbol{y} \in \mathbf{CNF}_i(\mathscr{D})$ aus $\boldsymbol{x} \xrightarrow[\mathscr{D}]{i} \boldsymbol{y}$ bereits $\mathscr{D}, \boldsymbol{y} \vDash \varphi$ folgt.

(c) $\boldsymbol{EF}_i\varphi =_{\text{Def}} \boldsymbol{E}\,(\mathbf{1}\ \boldsymbol{U}_i\ \varphi)$;
$\mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{EF}_i\varphi$ gdw. es ein $\boldsymbol{\xi} \in \mathbf{P}_{\mathscr{D},i}(\boldsymbol{x})$ und ein $j < |\boldsymbol{\xi}|$ mit $\mathscr{D}, \boldsymbol{\xi}(j) \vDash \varphi$ gibt.

(d) $\boldsymbol{AF}_i\varphi =_{\text{Def}} \boldsymbol{A}\,(\mathbf{1}\ \boldsymbol{U}_i\ \varphi)$;
$\mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{AF}_i\varphi$ gdw. es für alle Wege $\boldsymbol{\xi} \in \mathbf{P}_{\mathscr{D},i}(\boldsymbol{x})$ ein $j < |\boldsymbol{\xi}|$ mit $\mathscr{D}, \boldsymbol{\xi}(j) \vDash \varphi$ gibt.

(e) $\boldsymbol{EG}_i\varphi =_{\text{Def}} \neg\boldsymbol{AF}_i\neg\varphi$;
$\mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{EG}_i\varphi$ gdw. es einen $i$-Weg $\boldsymbol{\xi} \in \mathbf{P}_{\mathscr{D},i}(\boldsymbol{x})$ gibt, so daß $\mathscr{D}, \boldsymbol{\xi}(j) \vDash \varphi$ für alle $j$ mit $0 \leqslant j < |\boldsymbol{\xi}|$ gilt.

(f) $\boldsymbol{AG}_i\varphi =_{\text{Def}} \neg\boldsymbol{EF}_i\neg\varphi$;
$\mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{AG}_i\varphi$ gdw. $\mathscr{D}, \boldsymbol{\xi}(j) \vDash \varphi$ für alle $i$-Wege $\boldsymbol{\xi} \in \mathbf{P}_{\mathscr{D},i}(\boldsymbol{x})$ und für alle $j$ mit $0 \leqslant j < |\boldsymbol{\xi}|$ gilt.

□

**Abgeleitete Operatoren II.** Neben den „gewöhnlichen“ CTL-Ableitungen können in DCTL eine Reihe weiterer interessanter Operatoren definiert werden.

IV.1.17. Definition (Beschränkte Operatoren). Für $C \subseteq A_{\mathscr{D}}$ sei

$$\boldsymbol{E}\,[C]\,(\varphi\ \boldsymbol{U}_i\ \psi) \;=_{\text{Def}}\; \boldsymbol{E}\left(\left[\varphi \wedge \bigwedge_{a \in A_i - C} \neg\,\langle a\rangle_i\,\mathbf{1}\right]\ \boldsymbol{U}_i\ \psi\right), \text{ und}$$

$$\boldsymbol{A}\,[C]\,(\varphi\ \boldsymbol{U}_i\ \psi) \;=_{\text{Def}}\; \boldsymbol{A}\left(\left[\varphi \wedge \bigwedge_{a \in A_i - C} \neg\,\langle a\rangle_i\,\mathbf{1}\right]\ \boldsymbol{U}_i\ \psi\right).$$

Daraus ergeben sich die Operatoren

$$\begin{array}{llll} \boldsymbol{E}\,[C]\,\boldsymbol{F}_i\varphi & =_{\text{Def}} & \boldsymbol{E}\,[C]\,(\mathbf{1}\ \boldsymbol{U}_i\ \varphi), & \boldsymbol{A}\,[C]\,\boldsymbol{F}_i\varphi \;=_{\text{Def}}\; \boldsymbol{A}\,[C]\,(\mathbf{1}\ \boldsymbol{U}_i\ \varphi), \\ \boldsymbol{E}\,[C]\,\boldsymbol{G}_i\varphi & =_{\text{Def}} & \neg\boldsymbol{A}\,[C]\,\boldsymbol{F}_i\neg\varphi, & \boldsymbol{A}\,[C]\,\boldsymbol{G}_i\varphi \;=_{\text{Def}}\; \neg\boldsymbol{E}\,[C]\,\boldsymbol{F}_i\neg\varphi. \end{array}$$

□

Es gilt also $\boldsymbol{E}\,[C]\,(\varphi\ \boldsymbol{U}_i\ \psi)$, wenn der $i$-Weg, für den die Formel $\boldsymbol{E}\,(\varphi\ \boldsymbol{U}_i\ \psi)$ gilt, lediglich mit Hilfe von Schritten solcher Aktionen aus $C \cap A_i$ konstruiert wird: Bei keinem auf diesem Weg liegenden Zustand ist eine der Aktionen $b \in A_i - C$ konzessioniert.

IV.1.18. Definition. Setzen wir $C_1 =_{\text{Def}} A_1$ sowie $C_i =_{\text{Def}} A_i - \bigcup_{j<i} A_j$ für $1 < i \leqslant N$. Weitere interessante abgeleitete Operatoren sind

$$\begin{array}{lll} \boldsymbol{EF}(\varphi_1, \varphi_2, \ldots, \varphi_N) & =_{\text{Def}} & \boldsymbol{E}\,[C_1]\,\boldsymbol{F}_1(\varphi_1 \wedge \boldsymbol{E}\,[C_2]\,\boldsymbol{F}_2(\varphi_2 \wedge \cdots \boldsymbol{E}\,[C_N]\,\boldsymbol{F}_N\varphi_N)\cdots), \\ \boldsymbol{AF}(\varphi_1, \varphi_2, \ldots, \varphi_N) & =_{\text{Def}} & \boldsymbol{A}\,[C_1]\,\boldsymbol{F}_1(\varphi_1 \wedge \boldsymbol{A}\,[C_2]\,\boldsymbol{F}_2(\varphi_2 \wedge \cdots \boldsymbol{A}\,[C_N]\,\boldsymbol{F}_N\varphi_N)\cdots) \end{array}$$

□

$\boldsymbol{EF}(\varphi_1, \varphi_2, \ldots, \varphi_N)$ ist so zu lesen. Es gibt eine Verhaltensweise, in der der Agent 1 zunächst einen Zustand mit $\varphi_1$ erreicht, dann übernimmt der Agent 2 die Kontrolle über die quantorisierte Verhaltensweise und erreicht ohne jede Interaktion mit 1 einen Zustand mit $\varphi_2$ (aber 1 darf durchaus nebenläufig zu 2 Aktionen

ausführen). Danach übernimmt 3 die Kontrolle und erreicht wiederum ohne Interaktion mit 1 oder 2 einen Zustand mit $\varphi_3$. Diese Übergabe der Kontrolle des Agenten $i-1$ an den Agenten $i$ setzt sich bis zum „größten" Agenten $N$ fort.

Das folgende Lemma ist der Schlüssel zur Berechnung der $\boldsymbol{U}$-Operatoren: Beide Operatoren werden auf die iterierte Anwendung der $\boldsymbol{X}$-Operatoren zurückgeführt.

IV.1.19. Lemma. *Die folgenden Äquivalenzen sind erfüllt:*

$$\begin{aligned}\mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{E}(\varphi\ \boldsymbol{U}_i\ \psi) &\quad\Leftrightarrow\quad \mathscr{D}, \boldsymbol{x} \vDash \psi \vee (\varphi \wedge \boldsymbol{E}\boldsymbol{X}_i \boldsymbol{E}(\varphi\ \boldsymbol{U}_i\ \psi))\,,\\ \mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{A}(\varphi\ \boldsymbol{U}_i\ \psi) &\quad\Leftrightarrow\quad \mathscr{D}, \boldsymbol{x} \vDash \psi \vee (\varphi \wedge \boldsymbol{A}\boldsymbol{X}_i \boldsymbol{A}(\varphi\ \boldsymbol{U}_i\ \psi))\,.\end{aligned}$$

**CTL.** Wie ist nun das Verhältnis der verteilten Logik DCTL zu der über sequentiellen Modellen interpretierten Logik CTL? Konstruieren wir durch die folgenden Festlegungen aus dem verteilten System $\mathscr{D}$ über $\Sigma$ und $J$ ein weiteres verteiltes System $\mathscr{D}^{\mathrm{seq}}$ über $\Sigma^{\mathrm{seq}} =_{\mathrm{Def}} \langle A_\Sigma, \varnothing\rangle$ und $J^{\mathrm{seq}} =_{\mathrm{Def}} \{1\}$:

(a) $\mathscr{T}_{\mathscr{D}^{\mathrm{seq}}} = \mathscr{T}_{\mathscr{D}}$, jedoch ist $\mathscr{T}_{\mathscr{D}^{\mathrm{seq}}}$ ein Spursystem über $\Sigma^{\mathrm{seq}}$,
(b) $\boldsymbol{\Theta}_{\mathscr{D}^{\mathrm{seq}}} =_{\mathrm{Def}} \{\{p\}\}$
(c) $\theta_{\mathscr{D}^{\mathrm{seq}}}(s) = \{p\}$ für alle $s \in S_{\mathscr{D}^{\mathrm{seq}}}$ und $\theta_{\mathscr{D}^{\mathrm{seq}}}(a) = \langle\{p\},\{p\}\rangle$ für alle $a \in A_{\Sigma^{\mathrm{seq}}}$.

$\mathscr{D}^{\mathrm{seq}}$ weist also nur eine triviale Verteilung auf; es gibt nur einen einzigen Agenten 1. Man beachte, daß $\mathbf{CNF}_1(\mathscr{D}^{\mathrm{seq}}) = \mathbf{L}(\mathscr{D}^{\mathrm{seq}})$ gilt.

Die Logik CTL besteht nun aus Formeln der Menge $\Phi(P,\{A\})$ für eine Menge atomarer Aussagen $P$. Die Semantik von CTL-Formeln entspricht derjenigen von DCTL, allerdings für Modelle der Form $\mathscr{D}^{\mathrm{seq}}$, hierbei ist $A = A_{\Sigma^{\mathrm{seq}}}$. Wenn wir eine CTL-Formel schreiben wollen, lassen wir den Index 1 für den einzigen Agenten $1 \in J^{\mathrm{seq}}$ weg, ebenso den Suffix [1] bei atomaren Formeln. So sind etwa $p$ für $p \in P$, $\langle a\rangle\varphi$ und $[a]\varphi$ für $a \in A_{\Sigma^{\mathrm{seq}}}$, $\boldsymbol{EF}\varphi$ oder $\boldsymbol{A}(\varphi\ \boldsymbol{U}\ \psi)$ CTL-Formeln.

**Globale Systemeingenschaften.** Globale Systemeigenschaften, also solche Eigenschaften, die sich auf das nebenläufige Verhalten von mehr als einem Agenten beziehen, können nicht oder nur schwer in DCTL formuliert werden. Fassen wir den Begriff „globale Eigenschaft" zunächst etwas genauer. Erinnern wir uns, daß wir $\Theta_{\mathscr{D}} = \bigcup_{i\in J}\Theta_i$ gesetzt hatten, wobei $\boldsymbol{\Theta}_{\mathscr{D}} = \{\Theta_i\}_{i\in J}$ war. Wir nehmen nun an, daß $\Theta_{\mathscr{D}} \subseteq P$ für die Menge atomarer Aussagen $P$ ist, über der die Formelmenge $\Phi(P, \boldsymbol{A}(\mathscr{D}))$ definiert wird. Ist $\{\nu_i\}_{i\in J}$ eine Familie von Wertzuweisungen, so nehmen wir

$$(\theta_{\mathscr{D}}(\delta_{\mathscr{D}}(s_{\mathscr{D}}, x)) \cap \Theta_i) \subseteq \nu_i(\boldsymbol{x}) \qquad \text{(IV.1.}\alpha\text{)}$$

für alle $\boldsymbol{x} \in \mathbf{CNF}_i(\mathscr{D})$ an.

Weiterhin sei der Einfachheit halber für jede Position $p \in \Theta_{\mathscr{D}}$ eine Komplementärposition $p^{\mathrm{c}} \in \Theta_{\mathscr{D}}$; wir haben in Abschnitt II.1 bereits festgestellt, daß Komplementärpositionen leicht den Komponenten eines verteilten Systems hinzugefügt werden können.

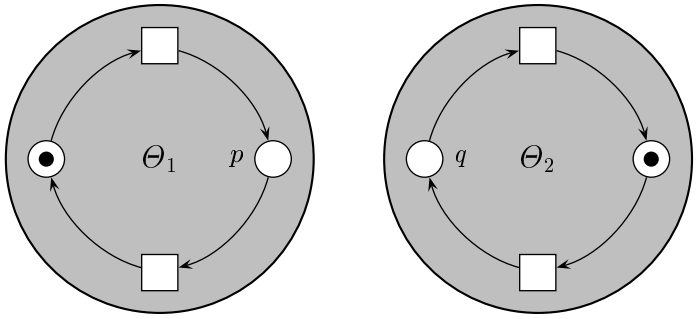



ABBILDUNG IV.5. Verteiltes System mit zwei unabhängigen Komponenten

IV.1.20. DEFINITION (Einfache globale Eigenschaft). Wir nennen jede Teilmenge $Q \subseteq \Theta_{\mathscr{D}}$ eine *einfache globale Eigenschaft* oder kurz eine *globale Eigenschaft*. Ist $s \in S_{\mathscr{D}}$ ein Zustand, so sagen wir, $s$ *erfüllt* $Q$, wenn $Q \subseteq \theta_{\mathscr{D}}(s)$ gilt. Ist $K = \{\beta_1, \beta_2, \ldots, \beta_k\} \subseteq J$ und $p_{\beta_i} \in \Theta_{\beta_i}$ für alle $\beta_i \in K$, so nennen wir die Formel

$$\tau = \bigwedge_{i=1}^{k} p_{\beta_i}$$

die *Spezifikation* der globalen Eigenschaft $Q = \{p_{\beta_1}, p_{\beta_2}, \ldots, p_{\beta_k}\}$. Ist $\tau$ die Spezifikation von $Q$ und $s \in S_{\mathscr{D}}$, so schreiben wir $\mathscr{D}, s \vDash \tau$, wenn $s$ $Q$ erfüllt. □

Natürlich gibt es globale Eigenschaften, die ebenfalls lokale Eigenschaften sind:

IV.1.21. BEISPIEL. Ist etwa

$$\tau = \bigwedge_{i=1}^{N} p_i \text{ mit } p_i \in \Theta_i,\ 1 \leqslant i \leqslant N,$$

und gibt es eine lokale Konfiguration $\boldsymbol{x} \in \mathbf{CNF}_N(\mathscr{D})$ mit

(a) $\mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{EF}(p_1[1], p_2[2], \ldots, p_N[N])$, so gibt es einen Zustand $s \in S_{\mathscr{D}}$ mit $\theta(s) = \{p_1, p_2, \ldots, p_N\}$, nämlich den Zustand $s = \delta_{\mathscr{D}}(s_{\mathscr{D}}, x)$; bzw.
(b) $\mathscr{D}, \boldsymbol{x} \vDash \boldsymbol{AF}(p_1[1], p_2[2], \ldots, p_N[N])$, so ist ein Zustand $s = \delta_{\mathscr{D}}(s_{\mathscr{D}}, x) \in S_{\mathscr{D}}$ mit $\theta(s) = \{p_1, p_2, \ldots, p_K\}$ unvermeidbar in $\mathscr{D}$.

Man beachte, daß diese Eigenschaften wegen Bedingung (IV.1.$\alpha$) nicht in der Form

$$\boldsymbol{EF}_N \bigwedge_{i=1}^{N} p[i] \text{ bzw. } \boldsymbol{AF}_N \bigwedge_{i=1}^{N} p[i]$$

ausgedrückt werden können. □

Allerdings existieren natürlich globale Eigenschaften, die nicht lokal sind; in Einzelfällen können jedoch auch solche Eigenschaften als Konjunktion lokaler Eigenschaften formuliert werden, wie das folgende Beispiel zeigt:

IV.1.22. Beispiel. Die Erreichbarkeit des Zustands $\{p, q\}$ in dem in Abb. IV.5 gezeigten Netzsystem (zwei Komponenten sind grau unterlegt) ist zwar in CTL als $\boldsymbol{EF}(p \wedge q)$, in DCTL jedoch nicht in der Form $\boldsymbol{EF}_1(p[1] \wedge q[2])$ oder $\boldsymbol{EF}_2(p[1] \wedge q[2])$ formulierbar. Allerdings kann die Aussage „$Q$ ist durch nebenläufiges Fortschreiten aller $N$ Agenten eines verteilten Systems von dessen Initialzustand aus erreichbar“ in der Form

$$\mathcal{D} \vDash \bigwedge_{i=1}^{N} \boldsymbol{E}\,[C^i]\ \boldsymbol{F}_i p_i[i] \text{ mit } C^i =_{\text{Def}} A_i - \bigcup\{A_j : i \neq j\}$$

mit $p_i \in \Theta_i$ formulierbar. Die Verwendung des eingeschränkten Operatores stellt sicher, daß der Weg des Agenten $i$, der $\boldsymbol{EF}_i p_i[i]$ erfüllt, nicht durch Aktionen von Agenten $j$ mit $i \neq j$ beeinflußt wird. □

Mit Hilfe von Kombinationen der in den Beispielen IV.1.21 und IV.1.22 beschriebenen Spezifikationstechniken für globale Eigenschaften kann die Erreichbarkeit bzw. Unvermeidbarkeit von Zuständen spezifiziert werden. Allerdings führen diese Spezifikationstechniken zu unübersichtlichen und schwer zu lesenden Formeln. Da zudem die Struktur einer solchen Formel auf die Abhängigkeiten der Agenten untereinander Bezug nimmt (etwa setzten die beiden Formeln in Beispiel IV.1.21 voraus, daß der $i$-te Agent eine direkte Kausalbeziehung mit dem $i+1$-ten Agenten unterhält), kann keine automatische Transformation einer Spezifikation $\tau$ einer globalen Eigenschaft in eine DCTL-Formel angegeben werden.

Globale Eigenschaften können aber zu lokalen gemacht werden, indem ein verteiltes System um einen Beobachter für eine solche globale Eigenschaft erweitert wird.

IV.1.23. Definition. Sei $K = \{\beta_1, \beta_2, \ldots, \beta_k\} \subseteq J$ und $p_{\beta_i} \in \Theta_{\beta_i}$ für alle $\beta_i \in K$. Sei

$$\tau = \bigwedge_{i=1}^{k} p_{\beta_i}$$

die Spezifikation einer globalen Eigenschaft $Q$. Ein *Beobachter* von $\tau$ ist eine Aktion $a_\tau \in A_\Sigma$ mit den folgenden Eigenschaften:

(a) $\theta_{\mathcal{D}}(a_\tau) = \theta^{\cdot}_{\mathcal{D}}(a_\tau) = Q$,
(b) $Q \subseteq \theta_{\mathcal{D}}(s) \Rightarrow s \xrightarrow[\mathcal{D}]{a_\tau}$ für alle $s \in S_{\mathcal{D}}$.

□

IV.1.24. Lemma. *Seien $K$, $\tau$ und $a_\tau$ wie in Def. IV.1.23. Dann gilt*

(a) $a_\tau \in A_{\beta_i}$ *für* $1 \leqslant i \leqslant k$,
(b) $p_{\beta_i} \in \dot{\theta}(a) \Rightarrow a\ D_\Sigma\ a_\tau$ *für alle* $a \in A_\Sigma$, $1 \leqslant i \leqslant k$.

Beweis. (a) gilt aufgrund von Eigenschaft (II.3.$\alpha$) in Def. II.3.2. (b) folgt dann nach Theorem II.3.3 aus (a). □

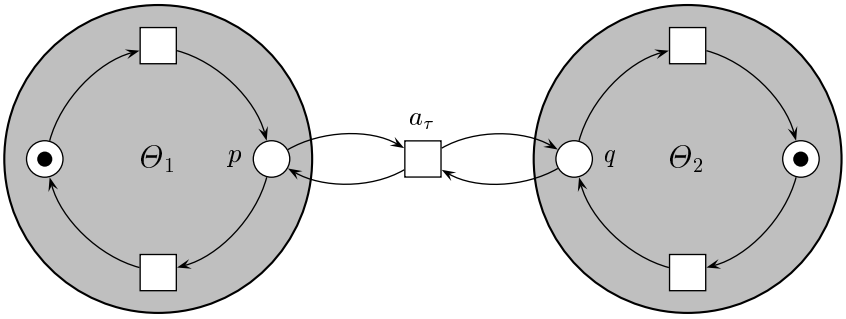



ABBILDUNG IV.6. Das verteilte System aus Abb. IV.5 mit einem Beobachter für $\tau = p \wedge q$.

---

IV.1.25. THEOREM. *Seien $K$, $\tau$ und $a_\tau$ wie in Def. IV.1.23. Weiterhin sei $\boldsymbol{x} \in \mathbf{LSSL}(\mathscr{D})$ und $s = \delta_{\mathscr{D}}(s_{\mathscr{D}}, x)$. Es gilt*

$$\mathscr{D}, s \vDash \tau \Leftrightarrow \bigwedge_{i=1}^{k} \left( \downarrow^{\beta_i} \boldsymbol{x} \vDash \langle a_\tau \rangle_{\beta_i} \mathbf{1} \right).$$

BEWEIS. ($\Rightarrow$) $\mathscr{D}, s \vDash \tau \Rightarrow Q \subseteq \theta_{\mathscr{D}}(s) \Rightarrow s \xrightarrow[\mathscr{D}]{a_\tau}$ nach Def. IV.1.23.(b). Da jedoch nach Lemma IV.1.24.(a) $a_\tau \in A_{\beta_i}$ für $1 \leqslant i \leqslant k$ gilt, folgt $\downarrow^{\beta_i} \boldsymbol{x} \xrightarrow[\mathscr{D},i]{a_\tau} \downarrow^{\beta_i} (\boldsymbol{x} \circ_\Sigma \boldsymbol{a}_\tau)$.

($\Leftarrow$) Offensichtlich. □

Für jede Spezifikation $\tau$ einer globalen Eigenschaft $Q$ kann also ein Beobachter $a_\tau$ zu einem verteilten System hinzugefügt werden; Def. IV.1.23 und Lemma IV.1.24 beschreiben, wie diese Transformation zu erfolgen hat. Das folgende Beispiel demonstriert das Hinzufügen eines Beobachters für Netzsysteme.

IV.1.26. BEISPIEL. Abb. IV.6 zeigt das verteilte System aus Abb. IV.5 zusammen mit einem Beobachter $a_\tau$ für die globale Eigenschaft $\tau = p \wedge q$. □

Man beachte: Das Hinzufügen eines Beobachters modifiziert die Kausalstruktur eines verteilten Systems. Da u. U. unabhängige Systemteile nun durch einen Beobachter in Beziehung gesetzt werden, können DCTL-Formeln, die für das nicht modifizierte System gegolten bzw. nicht gegolten haben, nun ihre Gültigkeit verlieren bzw. gewinnen.

## IV.2. Systemspezifikation mit DCTL — ein Beispiel

In diesem Abschnitt erläutern wir anhand eines Beispiels, wie die Spezifikation verteilter Systeme mit Hilfe der Logik DCTL erfolgen kann. Natürlich kann an dieser Stelle keine vollständige Fallstudie eines Systems nichttrivialer Komplexität durchgeführt werden, allerdings ist eine gewisse Komplexität des zu spezifizierenden Systems notwendig, um eine adäquate Bandbreite relevanter Eigenschaften zu

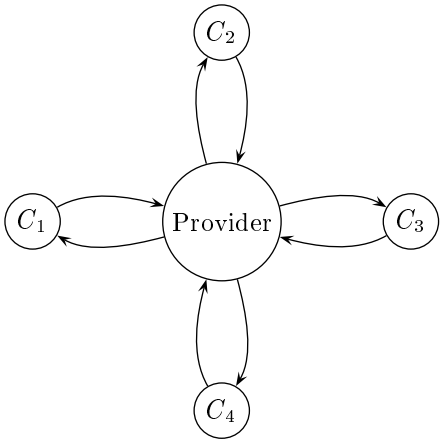



ABBILDUNG IV.7. Verteiltes Datenbanksystem

erhalten. Unsere Spezifikation kann deshalb nicht alle Aspekte unseres Beispielsystems abdecken.

**Ein verteiltes System.** Wir betrachten ein System bestehend aus $k$ ($k > 1$) als *Klienten* bezeichneten Komponenten $C_1, C_2, \ldots, C_k$ sowie einer weiteren Komponete $P$, die wir *Provider* nennen wollen. Die Klienten interagieren miteinander, indem sie den Provider als Medium verwenden, d. h. der Provider ist für den eigentlichen Datenaustausch verantwortlich; keiner der Klienten interagiert direkt mit irgendeinem anderen Klienten. Abb. IV.7 zeigt ein Schema der geschilderten Architektur.

Die Interaktion der Klienten und des Providers wird mit Hilfe eines *Kommunikationsprotokolls* (im folgenden kurz als *Protokoll* bezeichnet) beschrieben, das Kausalbeziehungen zwischen sog. *Dienstprimitiven* bestimmt. Dienstprimitive können als Grundfunktionalitäten eines Dienstgebers (hier des Providers) aufgefaßt werden. Auf der Beschreibungsebene, die wir in diesem Abschnitt verwenden, sind diese Dienstprimitive atomare Operationen (d. h. können als Aktionen in DCTL-Formeln verwendet werden). Moderne Protokolle sind jedoch durch eine Vielzahl von Schichten definiert; in tieferen Schichten der Protokollspezifikation kann eine detailiertere Beschreibungen dieser Primitive erfolgen.

Im *protocol engineering* ist es üblich, Dienstprimitiven *Typen* (im Sinne von Funktionen) zuzuordnen. Ist $\pi$ ein Dienstprimitiv, so bezeichnet $\pi.t$ die Funktionalität $t$ von $\pi$. Dabei beziehen sich diese Typen auf spezifische Interaktionen zwischen zwei Dienstnehmern. Derjenige Dienstnehmer, der eine solche Interaktion

einleitet, wird als *Initiator* bezeichnet, der andere Dienstnehmer ist der *Responder*. Übliche Typen von Dienstprimitiven sind:

*req* (*request*) Anfordern eines Dienstes durch einen der Dienstnehmer,
*ind* (*indication*) zugehörige Anzeige beim Partner der Interaktion,
*res* (*response*) Antwort auf eine Indikation,
*cnf* (*confirmation*) Bestätigung einer Diensterbringung.

Daneben können Dienstprimitive mit Parametern versehen werden. Solche Parameter umfassen üblicherweise eine Identifikation sowohl des Initiators als auch des Responders (die Adressen beider Agenten), eine Transaktionskennzeichnung und einen Datenanteil. Wir beschränken uns in unserem Beispiel auf Parameter $C_i$ und $C_j$, die den Initiator $C_i$ und den Responder $C_j$ einer Interaktion identifizieren, wir verwenden $i$ und $j$ als Indizes eines Dienstprimitivs, um anzuzeigen, daß es sich bei $C_i$ und $C_j$ um Parameter dieses Primitivs handelt. Damit bezeichnet $\pi.t_{i,j}$ den Typ $t$ des Diensprimitivs $\pi$, das mit den Adressen von $C_i$ als Initiator und $C_j$ als Responder parametrisiert wurde.

Die Dienstprimitive, die wir in unserem Beispiel verwenden, sind zusammen mit den zugehörigen Typen in der folgenden Tabelle aufgeführt:

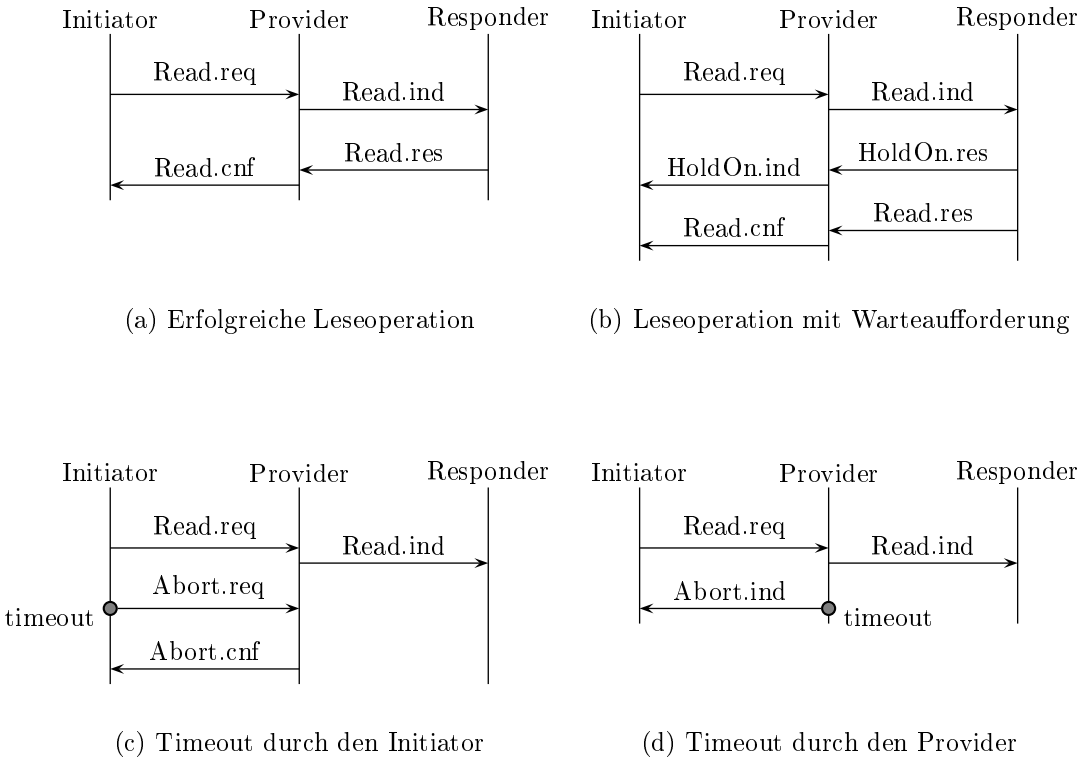
| Primitiv | Kürzel | *req* | *ind* | *res* | *cnf* |
|---|---|---|---|---|---|
| *Read* | *R* | × | × | × | × |
| *HoldOn* | *H* | × | × | | |
| *Abort* | *A* | × | × | | × |
| *Update* | *U* | × | × | × | × |

Die Einträge in der mit „Kürzel“ bezeichneten Spalte werden in DCTL-Formeln Verwendung finden. Im folgenden bezeichnen wir die Typisierung und Parametrisierung $\pi.t_{i,j}$ eines Dienstprimitivs $\pi$ ebenfalls als Dienstprimitiv.

Die Klienten in unserem Beispiel verwenden zwei Klassen von Interaktion, die *Leseoperation* und die *Aktualisierung*. Eine Leseoperation besteht in dem Anfordern von Daten von einem Klienten $C_j$ (dem Responder) durch einen Klienten $C_i$ (dem Initiator). Abb IV.8 stellt die möglichen Ausprägungen einer solchen Interaktion mit Hilfe sog. *Zeitdiagramme* dar.

IV.8.(a) erklärt die Interaktionsfolge für eine erfolgreiche Leseoperation. Der Initiator beginnt mit der Erzeugung des Primitivs *Read.req*. Die Anforderung des Initiators wird mit einem *Read.ind* dem Responder bekanntgegeben. Dieser antwortet mit einem *Read.res*, das als *Read.cnf* an den Initiator weitergeleitet wird. Wir gehen davon aus, daß nach der Erzeugung des *Read.req*-Primitivs durch den Initiator ein Timer gestartet wird, der die erlaubte Zeitspanne bis zu einer Antwort auf die Leseanforderung kennzeichnet.

IV.8.(b) stellt eine alternative Verhaltensweise dar: Sollte der Responder zum Zeitpunkt der Interaktion eine Aktualisierung vornehmen oder aus sonstigen Gründen zur Zeit nicht zur Durchführung einer Datenübermittlung an den Initiator




(a) Erfolgreiche Leseoperation

(b) Leseoperation mit Warteaufforderung

(c) Timeout durch den Initiator

(d) Timeout durch den Provider

ABBILDUNG IV.8. Zeitdiagramme für die Leseoperation

bereit sein, meldet er dies durch ein *HoldOn.req* an den Initiator. Das zugehörige *HoldOn.ind* beim Initiator veranlaßt diesen, den Timer zur Überwachung der Interaktion anzuhalten.

IV.8.(c) beschreibt den Fall, daß der durch den Timer des Initiators gegebene Zeitrahmen für die Antwort auf ein *Read.req*-Primitiv überschritten wird. In diesem Fall wird ein *Abort.req* erzeugt, das durch den Provider mit einem *Abort.cnf* beantwortet wird. *Abort.req* veranlaßt den Provider, keine (zu dieser Interaktion gehörenden) Nachrichten des Responders mehr an den Initiator weiterzuleiten.

IV.8.(d) Schließlich kann ein *Read.ind* beim Responder über eine bestimmte Zeitspanne hinaus unbeantwortet bleiben. Diese Zeitspanne ist durch einen weiteren Timer gegeben, der nach der Erzeugung des *Read.ind* durch den Provider gestartet wird. Nach Ablauf des Timers wird ein *Abort.cnf* beim Initiator erzeugt; jede Nachricht vom Responder wird ignoriert, bis eine neue Interaktion mit ihm stattfindet.

Bei der Aktualisierung handelt es sich um die Aufforderung eines Klienten $C_i$ an einen Klienten $C_j$, die bei ihm gelagerten Daten zu ändern. Wir verzichten darauf, auch für diese Interaktion Zeitdiagramme zu erfinden, legen aber fest, daß

eine Aktualisierung durch die Primitive *Update.req* beim Initiator und *Update.ind* bei Responder initiiert wird, während *Update.res* beim Responder und *Update.cnf* beim Initiator den erfolgreichen Abschluß der Interaktion kennzeichnen.

**Kopplung.** Wir legen fest, daß jeder der Klienten $C_i$ eine direkte Kausalbeziehung in Form der Dienstprimitive zu $P$ unterhält; d. h. die den Dienstprimitiven zugeordneten Systemaktionen befinden sich im Schnitt der Aktionsalphabete von $P$ und $C_i$. Etwa ist $\mathit{Read.req}_{i,j} \in A_P \cap A_{C_i}$ für alle $j \neq i$.

Dies ist eine sehr strenge und sicher unrealistische Annahme. I. Allg. sind Klienten und Provider nicht direkt miteinander gekoppelt, sondern verwenden ihrerseits zur Kommunikation ein in einer tieferen Schicht spezifiziertes Protokoll. Selbst wenn wir von dieser tieferen Schicht abstrahieren, findet ein Datenaustausch zwischen Klient und Provider jedoch nicht in Form eines synchronen *handshakes* statt, sondern verläuft asynchron. Allerdings ist der Aspekt einer asynchronen Kommunikation zweier Komponenten bereits dadurch erfaßt, daß der Provider als Medium einer solchen Kommunikation verwendet wird.

**Abgeleitete Operatoren III.** Neben den in Abschnitt IV.1 eingeführten abgeleiteten DCTL-Operatoren, die anscheinend in jedem Kontext nützlich sind, ergibt sich aus spezifischen Anwendungen oft die Notwendigkeit, spezielle Operatoren zu definieren, die den Anforderungen dieser Anwendung gerecht werden.

*Beschriftungen.* In der Praxis ist es häufig wünschenswert, unterschiedliche Systemaktionen mit demselben Namen zu bezeichnen. Etwa kann die Aktion *Read.cnf*, die die erfolgreiche Übertragung von Daten an den Initiator einer Leseoperation kennzeichnet, in zwei verschiedenen Kontexten ausgeführt werden, nämlich als Abschluß einer Interaktion ohne bzw. mit dem Stattfinden einer Warteaufforderung (vgl. Abb. IV.8, (a) und (b)).

Wir modellieren dies, indem wir eine Beschriftungsfunktion $\eta : A_\Sigma \to A$ verwenden, die Aktionen $a \in A_\Sigma$ eines Spuralphabets $\Sigma$ Symbole $\eta(A)$ aus einem weiteren Alphabet $A$ zuordnet. Allerdings gehen wir weiterhin davon aus, daß die betrachteten Spursysteme auch auf der Beschriftungsebene deterministisch sind, d. h. ist $\mathcal{T}$ ein Spursystem über $\Sigma$ und sind $a, b \in A_\Sigma$ mit $a\ D_\Sigma\ b$, so gilt für alle Zustände $s \in S_\mathcal{T}$: Wenn es Zustände $s_1, s_2 \in S_\mathcal{T}$ mit $s \xrightarrow[\mathcal{T}]{a} s_1$ und $s \xrightarrow[\mathcal{T}]{b} s_2$ gibt, so gilt $\eta(a) = \eta(b) \Rightarrow s_1 = s_2$.

Nehmen wir der Einfachheit halber an, eine solche Beschriftung $\eta : A_\Sigma \to A$ sei surjektiv. Nun ist es einfach, die aktionsbasierten DCTL-Operatoren auf Aktionen aus dem Alphabet $A$ zu definieren: Ist $\alpha \in A$, so setzen wir

$$\langle \alpha \rangle_i\, \varphi =_{\mathrm{Def}} \bigvee_{a \in \eta^{-1}(\alpha) \cap A_i} \langle a \rangle_i\, \varphi \text{ und } [\alpha]_i\, \varphi =_{\mathrm{Def}} \bigvee_{a \in \eta^{-1}(\alpha) \cap A_i} [a]_i\, \varphi.$$

In derselben Weise werden die beschränkten $\boldsymbol{U}$-Operatoren auf Teilmengen $C \subseteq A$ angepaßt:

$$\begin{aligned} \boldsymbol{E}\,[C]\,(\varphi\ \boldsymbol{U}_i\ \psi) &=_{\text{Def}} \boldsymbol{E}\,\left[\eta^{-1}(C) \cap A_i\right]\,(\varphi\ \boldsymbol{U}_i\ \psi) \text{ und} \\ \boldsymbol{A}\,[C]\,(\varphi\ \boldsymbol{U}_i\ \psi) &=_{\text{Def}} \boldsymbol{A}\,\left[\eta^{-1}(C) \cap A_i\right]\,(\varphi\ \boldsymbol{U}_i\ \psi)\,. \end{aligned}$$

Wir benötigen noch

$$\begin{aligned} \boldsymbol{E}\,[C]\,\boldsymbol{F}_i\varphi &=_{\text{Def}} \boldsymbol{E}\,\left[\eta^{-1}(C) \cap A_i\right]\,\boldsymbol{F}_i\varphi \text{ und} \\ \boldsymbol{A}\,[C]\,\boldsymbol{F}_i\varphi &=_{\text{Def}} \boldsymbol{A}\,\left[\eta^{-1}(C) \cap A_i\right]\,\boldsymbol{F}_i\varphi. \end{aligned}$$

Im folgenden schreiben wir $C_i$ sowohl für den Agenten $C_i$ als auch für das Aktionsalphabet $\eta(A_{C_i})$. Ebenso bezeichnet $P$ sowohl den Agenten $P$ wie auch das Alphabet $\eta(A_P)$.

*Bewirkt.* Die folgenden beiden Operatoren ermöglichen es uns, Aussagen der Form „Das Ereignis $\alpha_0$ bei dem Agenten $i$ bewirkt ein Ereignis $\alpha_0$ bei dem Agenten $j$, wobei die Kausalkette, die zum Stattfinden von $\alpha_1$ führt, über Ereignisse des Agenten $k$ verläuft“:

$$\begin{aligned} \alpha\ \boldsymbol{at}\ i\ \varphi &=_{\text{Def}} \langle\alpha\rangle_i\,\varphi \quad \text{bzw.} \quad \alpha\ \boldsymbol{at}\ i\ =_{\text{Def}} \langle\alpha\rangle_i\ \mathbf{1}, \\ \overset{k}{\underset{\boldsymbol{E}}{\Longmapsto}}\ \varphi &=_{\text{Def}} \boldsymbol{E}\boldsymbol{F}_k\varphi \quad \text{und} \quad \overset{k}{\underset{\boldsymbol{A}}{\Longmapsto}}\ \varphi =_{\text{Def}} \boldsymbol{A}\boldsymbol{F}_k\varphi. \end{aligned}$$

Eine Aussage der obigen Form kann nun mit Hilfe der Formeln

$$\alpha_0\ \boldsymbol{at}\ i \overset{k}{\underset{\boldsymbol{E}}{\Longmapsto}} \alpha_1\ \boldsymbol{at}\ j \text{ bzw. } \alpha_0\ \boldsymbol{at}\ i \overset{k}{\underset{\boldsymbol{A}}{\Longmapsto}} \alpha_1\ \boldsymbol{at}\ j.$$

formuliert werden. Sowohl $\alpha\ \boldsymbol{at}\ i$ als auch $\underset{\boldsymbol{E}}{\Longmapsto}$ und $\underset{\boldsymbol{A}}{\Longmapsto}$ sind unäre Operatoren und binden damit stärker als die binären DCTL-Operatoren.

Wir verwenden auch die beschränkten Versionen der $\Mapsto$-Operatoren, nämlich

$$\overset{k}{\underset{\boldsymbol{E}[C]}{\Longmapsto}}\ \chi =_{\text{Def}} \boldsymbol{E}\,[C]\,\boldsymbol{F}_k\chi \text{ und } \overset{k}{\underset{\boldsymbol{A}[C]}{\Longmapsto}}\ \chi =_{\text{Def}} \boldsymbol{A}\,[C]\,\boldsymbol{F}_k\chi.$$

Man beachte, daß Formeln der Form $\alpha\ \boldsymbol{at}\ i \overset{j}{\underset{\boldsymbol{E}}{\Longmapsto}} \varphi$ bzw. $\alpha\ \boldsymbol{at}\ i \overset{j}{\underset{\boldsymbol{A}}{\Longmapsto}} \varphi$ für Agenten $i$ und $j$ mit $i \neq j$ nicht in CTL ausgedrückt werden können. Abb. IV.9 zeigt zwei verteilte Systeme in Netzsyntax; die Aktionsalphabete seien dabei

$$\begin{aligned} \mathscr{D}_1 &: A_1 = \{a\}, A_2 = \{b\} \\ \mathscr{D}_2 &: A_1 = \{a\}, A_2 = \{a, b\}. \end{aligned}$$

In der DCTL-Version über der angegebenen Komponentenüberdeckung ist $\mathscr{D}_1 \models a\ \boldsymbol{at}\ 1 \overset{2}{\underset{\boldsymbol{A}}{\Longmapsto}} b\ \boldsymbol{at}\ 2$ falsch, während $\mathscr{D}_2 \models a\ \boldsymbol{at}\ 1 \overset{2}{\underset{\boldsymbol{A}}{\Longmapsto}} b\ \boldsymbol{at}\ 2$ wahr ist.

Es ist leicht zu sehen, daß die durch die beiden Netzsysteme induzierten Transitionssysteme isomorph sind: Es gilt $\mathscr{D}_1^{\text{seq}} \models \varphi \Leftrightarrow \mathscr{D}_2^{\text{seq}} \models \varphi$ für alle CTL-Formeln $\varphi$ über einer passenden Menge $P$ atomarer Aussagen.

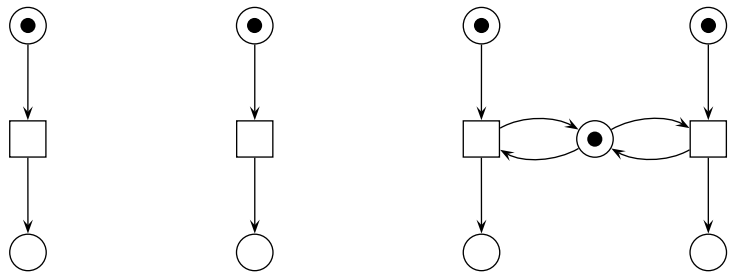



ABBILDUNG IV.9. Zwei verteilte Systeme in Netzsyntax.

**Allgemeine Eigenschaften.** Es soll weder den Klienten unseren verteilten Systems noch dem Provider erlaubt sein zu terminieren. Diese Aussage kann leicht in DCTL formuliert werden:

$$\bigwedge_{i=1}^{k} \boldsymbol{AG}_{C_i} \boldsymbol{EX}_{C_i} \mathbf{1} \wedge \boldsymbol{AG}_P \boldsymbol{EX}_P \mathbf{1}.$$

Für den Provider wollen wir darüber hinaus „Livelockfreiheit“ fordern. Livelockfreiheit bedeutet, daß der Provider nicht in der Lage ist, eine unendliche Folge *interner* Aktionen (d.h. solche Aktionen, die die nicht zu Dienstprimitiven gehören) abzuarbeiten. Anders gesagt: Die Abarbeitung jedes Dienstprimitivs terminiert. Dazu sei $\Pi$ die Menge der durch $P$ zur Verfügung gestellten Dienstprimitive.

$$\boldsymbol{AG}_P \boldsymbol{AF}_P \left( \bigvee_{\pi \in \Pi} \pi \ \boldsymbol{at}\ P \right).$$

Schließlich fordern wir, daß zwischen einem Agenten $C_i$ und einem Agenten $C_j$ zu jedem Zeitpunkt eine Verbindung möglich sein soll:

$$\boldsymbol{AG}_{C_i} \left\{ R.req_{i,j} \ \boldsymbol{at}\ C_i \supset R.req_{i,j} \ \boldsymbol{at}\ C_i \overset{P}{\underset{\boldsymbol{E}}{\Longmapsto}} R.ind_{i,j} \ \boldsymbol{at}\ C_j \right\},$$
$$\boldsymbol{AG}_{C_i} \left\{ U.req_{i,j} \ \boldsymbol{at}\ C_i \supset U.req_{i,j} \ \boldsymbol{at}\ C_i \overset{P}{\underset{\boldsymbol{E}}{\Longmapsto}} U.ind_{i,j} \ \boldsymbol{at}\ C_j \right\}.$$

Wir können auch alternative Anforderungen an die Existenz einer Verbindung stellen: Ein Verbindungsaufbau zwischen $C_i$ und $C_j$ ist möglich, ohne daß einer der anderen Agenten $C_l$ diesen beeinflußt: Damit ist sichergestellt, daß tatsächlich eine Verbindung zwischen $C_i$ und $C_j$ existiert und nicht nur eine fehlerhaft adressierte

Nachricht eines anderen Agenten $C_l$ an $C_j$ zugestellt wurde. Sei dazu

$$C = \bigcup\{C_l : 1 \leqslant l \leqslant n \;\&\; i \neq l \;\&\; j \neq l\}.$$

Die beschriebene Eigenschaft ist dann für die Leseoperation als

$$\boldsymbol{EF}_{C_i}\left\{R.req_{i,j}\ \boldsymbol{at}\ C_i \supset R.req_{i,j}\ \boldsymbol{at}\ C_i \overset{P}{\underset{\boldsymbol{E}[C]}{\Longmapsto}} R.ind_{i,j}\ \boldsymbol{at}\ C_j\right\}$$

in DCTL formulierbar. Man beachte, daß der $\boldsymbol{AG}$-Operator an dieser Stelle nicht anwendbar ist, da $C_j$ und $C_l$ natürlich eine Interaktion unterhalten dürfen.

**Leseoperation.** Für jeden Klienten sowie für den Provider ist zunächst die Reihenfolge der bei ihnen auftretenden Dienstprimitive zu klären. Die entsprechenden Formeln können direkt aus den Zeitdiagrammen in Abb. IV.8 abgelesen werden. Ist $C_i$ etwa der Initiator und $C_j$ der Responder in einer Leseinteraktion, so ergibt sich

$$\boldsymbol{AG}_{C_i}\Big\{R.req_{i,j}\ \boldsymbol{at}\ C_i \supset R.req_{i,j}\ \boldsymbol{at}\ C_i \overset{C_i}{\underset{\boldsymbol{A}}{\Longmapsto}} \big(\quad R.cnf_{i,j}\ \boldsymbol{at}\ C_i \vee H.ind_{i,j}\ \boldsymbol{at}\ C_i \vee A.cnf_{i,j}\ \boldsymbol{at}\ C_i \vee A.ind_{i,j}\ \boldsymbol{at}\ C_i\big)\Big\}.$$

D. h. das Auftreten des Dienstprimitivs *Read.req* bei $C_i$ hat zur Folge, daß bei $C_i$ irgendwann eines der Dienstprimitive *Read.req HoldOn.ind*, *Abort.ind* oder *Abort.cnf* auftritt. Ebenso führt *HoldOn.ind* zu *Read.cnf* und *Abort.req* zu *Abort.cnf*.

$$\boldsymbol{AG}_{C_i}\big\{H.ind_{i,j}\ \boldsymbol{at}\ C_i \supset H.ind_{i,j}\ \boldsymbol{at}\ C_i \overset{C_i}{\underset{\boldsymbol{A}}{\Longmapsto}} R.cnf_{i,j}\ \boldsymbol{at}\ C_i\big\},$$

$$\boldsymbol{AG}_{C_i}\left\{A.req_{i,j}\ \boldsymbol{at}\ C_i \supset A.req_{i,j}\ \boldsymbol{at}\ C_i \overset{C_i}{\underset{\boldsymbol{A}}{\Longmapsto}} A.cnf_{i,j}\ \boldsymbol{at}\ C_i\right\}.$$

Schließlich ist noch das Verhältnis eines Timeouts bei $C_i$ und dem zugehörigen *Abort.req* bei $C_i$ zu klären. Wir übersetzen das durch Zeitdiagramm IV.8.(c) implizierte „sofort“ mit „im nächsten Schritt“:

$$\boldsymbol{AG}_{C_i}\left\{timeout\ \boldsymbol{at}\ C_i \supset timeout\ \boldsymbol{at}\ C_i\ A.req\ \boldsymbol{at}\ C_i\right\}.$$

Die Diagrammteile, die sich auf den Provider und den Responder beziehen, können in derselben Weise umgesetzt werden.

Wir spezifizieren nun die Interaktion der an einer Leseoperation beteiligten Agenten. $C_i$ sei der Initiator, $C_j$ der Responder; wir beschränken uns auf die Diagramme IV.8.(a) und IV.8.(b).

$$\boldsymbol{AG}_{C_i}\left\{R.req_{i,j}\ \boldsymbol{at}\ C_i \supset R.req_{i,j}\ \boldsymbol{at}\ C_i \overset{P}{\underset{\boldsymbol{A}}{\Longmapsto}} R.ind_{i,j}\ \boldsymbol{at}\ C_j\right\},$$

$$\boldsymbol{AG}_{C_j}\left\{R.res_{i,j}\ \boldsymbol{at}\ C_j \supset R.res_{i,j}\ \boldsymbol{at}\ C_j \overset{P}{\underset{\boldsymbol{A}}{\Longmapsto}} R.cnf_{i,j}\ \boldsymbol{at}\ C_i\right\},$$

$$\boldsymbol{AG}_{C_j}\left\{H.req_{i,j}\ \boldsymbol{at}\ C_j \supset H.req_{i,j}\ \boldsymbol{at}\ C_j \overset{P}{\underset{\boldsymbol{A}}{\Longmapsto}} H.ind_{i,j}\ \boldsymbol{at}\ C_i\right\}.$$

Schließlich soll sichergestellt sein, daß eine neue Leseoperation zwischen zwei Klienten erst dann eingeleitet werden kann, wenn alle vorhergehenden Leseoperationen abgeschlossen sind. Diese Eigenschaft wird sowohl aus Sicht des Initiators $C_i$ als auch aus Sicht des Responders $C_j$ spezifiziert:

$$\boldsymbol{AG}_{C_i}\left\{R.req_{i,j} \supset R.req_{i,j} \stackrel{C_i}{\Longmapsto\!\!\!\Longrightarrow}_{\boldsymbol{A}[C_i-\{R.req_{i,j}\}]} R.cnf_{i,j}\right\},$$

$$\boldsymbol{AG}_{C_j}\left\{R.ind_{i,j} \supset R.ind_{i,j} \stackrel{C_j}{\Longmapsto\!\!\!\Longrightarrow}_{\boldsymbol{A}[C_j-\{R.ind_{i,j}\}]} R.res_{i,j}\right\}.$$

Allerdings soll erlaubt sein, daß mehr als ein Klient lesenden Zugriff auf denselben Responder erhält. Wenn wir etwa fordern wollen, daß der Klient $C_1$ Leseanforderungen der anderen Klienten $C_2, C_2, \ldots, C_k$ (der Einfachheit halber in dieser Reihenfolge) entgegennehmen kann, bevor irgendeine dieser Anforderungen bearbeitet wird (d. h. die Resourcen des Klienten $C_1$ sind ausreichend zur Behandlung der maximal möglichen Anzahl von Leseanforderungen), würden wir unter Verwendung der Festlegung

$$C =_{\mathrm{Def}} C_1 - \left(\bigcup_{i=2}^{k}\{R.res_{i,1}\} \cup \bigcup_{i=2}^{k}\{H.req_{i,1}\}\right)$$

spezifizieren:

$$\boldsymbol{EF}_1\left\{R.req_{2,1} \stackrel{C_1}{\Longmapsto\!\!\!\Longrightarrow}_{\boldsymbol{E}[C]} R.req_{3,1} \stackrel{C_1}{\Longmapsto\!\!\!\Longrightarrow}_{\boldsymbol{E}[C]} \cdots \stackrel{C_1}{\Longmapsto\!\!\!\Longrightarrow}_{\boldsymbol{E}[C]} R.req_{k,1}\right\}.$$

**Datenaktualisierung.** Wieder sei $C_i$ der Initiator und $C_j$ der Responder der Interaktion. Wir fordern, daß keine Leseaufforderungen beantwortet werden sollen, während eine Datenaktualisierung stattfindet:

$$\boldsymbol{AG}_{C_j}\left\{U.req_{i,j}\ \boldsymbol{at}\ C_j \supset \bigwedge_{i=1}^{k} \boldsymbol{A}\left(\neg R.res_{k,j}\ \boldsymbol{at}\ C_j\ \boldsymbol{U}_{C_j}\ U.res_{i,j}\ \boldsymbol{at}\ C_j\right)\right\}$$

Allerdings soll ein während einer Aktualisierung auftretendes *Read.ind* mit einem *HoldOn.req* beantwortet werden: Setzen wir

$$C^k \ =_{\mathrm{Def}}\ C_j - \left(\{U.ind_{j,i}, U.res_{i,j}\} \cup \bigcup_{i=1}^{k}\{R.ind_{k,j}, H.req_{k,j}\}\right),$$

dann ist diese Eigenschaft als

$$\boldsymbol{AG}_j\Bigg\{U.ind_{i,j}\ \boldsymbol{at}\ C_j \supset U.ind_{i,j}\ \boldsymbol{at}\ C_j \stackrel{C_j}{\Longmapsto\!\!\!\Longrightarrow}_{\boldsymbol{A}[C]} \left[U.res_{i,j}\ \boldsymbol{at}\ C_j \vee \bigvee_{i=1}^{k}\left(R.ind_{k,j}\ \boldsymbol{at}\ C_j \stackrel{C_j}{\Longmapsto\!\!\!\Longrightarrow}_{\boldsymbol{A}[C^k]} H.req_{k,j}\ \boldsymbol{at}\ C_j \stackrel{C_j}{\Longmapsto\!\!\!\Longrightarrow}_{\boldsymbol{A}[C^k]} U.res_{i,j}\right)\right]\Bigg\}$$

in DCTL formulierbar.

**Shutdown.** Um schließlich die Verwendung globaler Eigenschaften zu demonstrieren, führen wir ein weiteres Dienstprimitiv *Shutdown* (abgekürzt $S$) mit dem einzigen Typ *ind* und einem einzelnen Adressparameter $i$ ein, mit dessen Hilfe der Provider alle Klienten $C_i$ von einer (einstweiligen) Aussetzung seiner Dienstbereitstellung informiert. Der Klient $C_i$ nimmt dann einen Wartezustand $idle_i$ ein:

$$\boldsymbol{AG}_P \left\{ S.ind_1 \supset S.ind_1 \ \boldsymbol{at}\ P \ S.ind_2 \ \boldsymbol{at}\ P \ \ldots S.ind_k \ \boldsymbol{at}\ P \ \boldsymbol{AF}_{C_k} \left( \bigwedge_{i=1}^{k} idle_i \right) \right\}$$

Hierbei haben wir vorausgesetzt, daß der Provider die Shutdown-Benachrichtigungen in der Reihenfolge $C_1, C_2, \ldots, C_k$ sendet, und daß in jedem Schritt eine Benachrichtigung gesendet wird.

Diese konkretere Eigenschaft zeigt im Gegensatz zu den in Abschnitt IV.1 zur Illustration globaler Eigenschaften diskutierten Beispielen, daß die Verwendung globaler Eigenschaften zur Spezifikation verteilten Verhaltens äußerst irreführend sein kann. In Abhängigkeit von der Geschwindigkeit des Datenausstausches zwischen Provider und Klienten kann die Shutdown-Benachrichtigung einen Klienten $C_i$ erst erreichen, wenn der Provider den Betrieb bereits wieder aufgenommen hat, während ein anderer Klient $C_j$ sofort von der Aussetzung und der Wiederaufnahme der Diensterbringung benachrichtigt wird. In diesem Fall existiert ein Zustand nicht, in dem sich beide Klienten in ihrem Wartezustand befinden.

In DCTL kann diese Eigenschaft allerdings ohne die Annahme eines globalen Zustands formuliert werden, in dem alle Klienten auf eine Wiederaufnahme des Betriebs des Providers warten:

$$\begin{aligned}\boldsymbol{AG}_P \Big\{ & S.ind_1 \ \boldsymbol{at}\ P \supset S.ind_1 \ \boldsymbol{at}\ P \\ & \left( \overset{C_1}{\underset{\boldsymbol{A}[C_1-P]}{\Longrightarrow}} idle_1 \wedge S.ind_2 \ \boldsymbol{at}\ P \right. \\ & \quad \left( \overset{C_2}{\underset{\boldsymbol{A}[C_2-P]}{\Longrightarrow}} idle_2 \wedge S.ind_3 \ \boldsymbol{at}\ P \right. \\ & \qquad \left. \left. \left( \cdots \left( S.ind_k \ \boldsymbol{at}\ P \overset{C_k}{\underset{\boldsymbol{A}[C_k-P]}{\Longrightarrow}} idle_k \right) \cdots \right) \right) \right) \Big\}.\end{aligned}$$

Die Formel besagt, daß nach der Benachrichtung des Agenten $i$ dieser den Zustand $idle_i$ erreicht, ohne daß es dabei zu einer weiteren Interaktion mit dem Provider kommt.

## IV.3. Ein Modelchecker für DCTL

Unter einem *Modelchecker* für eine Logik $L$ (eine formale Sprache mit einer formal bestimmten Semantik, die Wörtern dieser Sprache Wahrheitswerte zuordnet) verstehen wir einen Algorithmus (oder seine Implementierung), der für eine gegebene Struktur $\mathcal{S}$, die den Anforderungen an ein Modell für eine Formel $\varphi$ aus

$L$ genügt, entscheidet, ob $\mathcal{S}$ ein Modell für $\varphi$ ist, wobei natürlich die Bedeutung des Begriffs „Modell“ von der Definition des Begriffs der „Gültigkeit“ für die Logik $L$ abhängig ist.

Das Erfüllbarkeitsproblem für $L$ ist hingegen allgemeiner formuliert: Eine Formel $\varphi$ aus $L$ ist erfüllbar, wenn es ein Modell $\mathcal{S}$ für $\varphi$ gibt, man fragt hier also: „Gibt es ein Modell $\mathcal{S}$ für $\varphi$?“ Betrachtet man temporale Logiken, deren Modelle Relationen über den Zuständen eines Systems sind, ergibt sich aus einer Lösung des Erfüllbarkeitsproblems (die die Konstruktion des Modells $\mathcal{S}$ beinhaltet) i. d. R. ein Modelchecker. Alle gebräuchlichen Techniken zum Modelchecking für temporale Logiken für lineare Zeitmodelle beruhen auf diesem Ansatz. Allerdings gibt es temporale Logiken, für die das Erfüllbarkeitsproblem unentscheidbar ist. Dennoch kann für solche Logiken häufig ein Modelchecker gefunden werden.

In diesem Abschnitt beschreiben wir einen Modelchecker für DCTL, der Prozeßautomaten $\mathcal{A}$ als Datenstruktur zur Beschreibung des Verhaltens eines verteilten Systems verwendet.

**Vorbereitende Annahmen.** Zunächst machen wir einige syntaktische Annahmen über Prozeßautomaten, deren Einhaltung jedoch durch die Algorithmen III.1 oder III.3 leicht gewährleistet werden kann.

Sei $\mathcal{A}$ ein Prozeßautomat zu einem verteilten System $\mathcal{D}$ (d. h. zu dem Spursystem $\mathcal{T}_{\mathcal{D}}$) über $\Sigma$ und $J$. Wir nehmen an, daß die Ereignismengen aller in $X_{\mathcal{A}}$ vorkommenden Semiordnungen paarweise disjunkt sind. Damit können wir definieren:

$$E_{\mathcal{A}} =_{\text{Def}} \bigcup_{x \in X_{\mathcal{A}}} E_x, <_{\mathcal{A}} =_{\text{Def}} \bigcup_{x \in X_{\mathcal{A}}} <_x \text{ sowie } \lambda_{\mathcal{A}} =_{\text{Def}} \bigcup_{x \in X_{\mathcal{A}}} \lambda_x.$$

Weiterhin ist die Operation $x_{\mathcal{A}} : E_{\mathcal{A}} \to X_{\mathcal{A}}$ mit

$$x_{\mathcal{A}}(e) =_{\text{Def}} \text{dasjenige } x \in X_{\mathcal{A}} \text{ mit } e \in E_x$$

wohldefiniert.

Zur weiteren Vereinfachung nehmen wir an, daß jede Semiordnung $x \in X_{\mathcal{A}}$ eindeutig die Zustände $s$ und $s'$ bestimmt, für die ein $x$-Übergang in $\mathcal{A}$ definiert ist. Für alle $x \in X_{\mathcal{A}}$ gilt also

$$\delta_{\mathcal{A}}(s_1, x) = s_1' \text{ definiert } \& \ \delta_{\mathcal{A}}(s_2, x) = s_2' \text{ definiert } \Rightarrow s_1 = s_2 \ \& \ s_1' = s_2'.$$

Das ist möglich, da $X_{\mathcal{A}}$ durchaus isomorphe Kopien von $x$ enthalten darf. Schließlich nehmen wir an, daß es für alle $x \in X_{\mathcal{A}}$ einen (eindeutig bestimmten) Zustand $s \in S_{\mathcal{A}}$ gibt, so daß $\delta_{\mathcal{A}}(s, x)$ definiert ist. Unter diesen beiden Annahmen sind die beiden Operationen $\lfloor \cdot \rfloor_{\mathcal{A}}, \lceil \cdot \rceil_{\mathcal{A}} : X_{\mathcal{A}} \to S_{\mathcal{A}}$ mit

$$\begin{aligned} \lfloor x \rfloor_{\mathcal{A}} &=_{\text{Def}} \text{derjenige Zustand } s \in S_{\mathcal{A}}, \text{ so daß } \delta_{\mathcal{A}}(s, x) \text{ definiert ist,} \\ \lceil x \rceil_{\mathcal{A}} &=_{\text{Def}} \text{derjenige Zustand } s' \in S_{\mathcal{A}}, \text{ so daß } \delta_{\mathcal{A}}(s, x) = s' \text{ definiert ist} \end{aligned}$$

wohldefinierte totale Funktionen.

**Relationen.** Mit Lemma IV.1.19 haben wir bereits festgestellt, daß die Berechnung der $\boldsymbol{U}$-Operatoren auf der iterierten Anwendung der $\boldsymbol{X}$-Operatoren beruht, diese wiederum werden auf den unbedingten Schrittoperator $\langle a \rangle_i$ zurückgeführt. Zur algorithmischen Behandlung des $\langle a \rangle_i$-Operators schließlich wird die auf $i$-lokalen Konfiguration definierte $i$-Schrittrelation benötigt. Zur Implementierung dieser Relation benötigen wir ein Analogon, das auf der Menge $E_{\mathscr{A}}$ operiert.

IV.3.1. Definition ($i$-Schrittrelation in $\mathscr{A}$). Sei für ein $a \in A_i$ die Relation $\xrightarrow[\mathscr{A},i]{a} \subseteq E_{\mathscr{A}} \times E_{\mathscr{A}}$ wie folgt definiert:

Es gilt $e_1 \xrightarrow[\mathscr{A},i]{a} e_2$ gdw. $\lambda_{\mathscr{A}}(e_1) \in A_i$ und $\lambda_{\mathscr{A}}(e_2) = a \in A_i$ vorliegen und eine der beiden folgenden Bedingungen erfüllt ist:

(a) Ist $x_{\mathscr{A}}(e_1) = x_{\mathscr{A}}(e_2)$, so gelte

$$e_1 <_{\mathscr{A}} e_2 \;\&\; \forall e \in E_{\mathscr{A}}\,(e_1 <_{\mathscr{A}} e <_{\mathscr{A}} e_2 \Rightarrow \lambda_{\mathscr{A}}(e) \notin A_i)\,,$$

(b) Liegt hingegen $x_{\mathscr{A}}(e_1) \neq x_{\mathscr{A}}(e_2)$ vor, so gilt

$$e_1 \in \max\nolimits_{\leqslant_{\mathscr{A}}} \left(E_{x_{\mathscr{A}}(e_1)} \cap \lambda_{\mathscr{A}}^{-1}(A_i)\right) \;\&\; e_2 \in \min\nolimits_{\leqslant_{\mathscr{A}}} \left(E_{x_{\mathscr{A}}(e_2)} \cap \lambda_{\mathscr{A}}^{-1}(A_i)\right)$$
$$\&\; \exists y_1 y_2 \ldots y_n \in \mathbf{P}_{\mathscr{A}}(\lceil x_{\mathscr{A}}(e_1)\rceil_{\mathscr{A}}, \lfloor x_{\mathscr{A}}(e_2)\rfloor_{\mathscr{A}}) \left(\bigcup_{j=1}^{n} E_{y_j} \cap \lambda_{\mathscr{A}}^{-1}(A_i) = \varnothing\right).$$

□

Um den Zusammenhang zwischen $\xrightarrow[\mathscr{D},i]{a}$ und $\xrightarrow[\mathscr{A},i]{a}$ zeigen zu können, definieren wir:

IV.3.2. Definition. Sei $e \in E_{\mathscr{A}}$. Die Menge der lokalen Konfigurationen, die bei $e$ akzeptiert werden, ist durch

$$\downarrow^i \boldsymbol{x} \in \mathbf{CNF}_{\mathscr{A}}(e)$$
$$\Leftrightarrow_{\text{Def}} \quad \exists \chi \in \mathbf{P}_{\mathscr{A}}(s_{\mathscr{A}}, \lfloor x_{\mathscr{A}}(e)\rfloor_{\mathscr{A}}) \left(x \equiv \mathrm{con}_{\Sigma}(\chi) \circ_{\Sigma} \left(x_{\mathscr{A}}(e)\left[\leqslant_{\mathscr{A}}^{-1}(e)\right]\right)\right)$$

definiert. □

Ist $\mathscr{A}$ ein vollständiger Prozeßautomat zu $\mathscr{D}$, so gibt es offenbar für alle $\boldsymbol{x} \in \mathbf{CNF}_i(\mathscr{D})$ ein $e \in E_{\mathscr{A}}$ mit $\lambda_{\mathscr{A}}(e) \in A_i$, so daß $\boldsymbol{x} \in \mathbf{CNF}_{\mathscr{A}}(e)$ ist. Allerdings kann es mehr als ein solches Ereignis geben.

IV.3.3. Definition. Die Menge derjenigen Ereignisse $e \in E_{\mathscr{A}}$, bei denen eine lokale Konfiguration $\boldsymbol{x} \in \mathbf{CNF}_i(\mathscr{D})$ akzeptiert wird, ist durch

$$\mathbf{AC}_{\mathscr{A},i}(\boldsymbol{x}) =_{\text{Def}} \{e \in E_{\mathscr{A}} : \lambda_{\mathscr{A}}(e) \in A_i \;\&\; \boldsymbol{x} \in \mathbf{CNF}_{\mathscr{A}}(e)\}$$

gegeben. □

IV.3.4. Lemma. *Sei $\mathscr{A}$ ein vollständiger Prozeßautomat zu $\mathscr{D}$. Sei $a \in A_\Sigma$, $e_1, e_2 \in E_{\mathscr{A}}$ und $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{CNF}_i(\mathscr{D})$.*

**algorithm** DCTL **is**
  **input** $\mathscr{A}$, ein Prozeßautomat zu einem
    wohlgeformten verteilten System $\mathscr{D}$ über $\Sigma$ und $J$;
    $\{\nu_i : E_{\mathscr{A}} \to \mathscr{P}(P)\}_{i\in J}$, eine Familie von Wertfunktionen;
    $\varphi : \Phi(P, \boldsymbol{A}(\mathscr{D}))$; eine DCTL-Formel;
  **output** true gdw. $\mathscr{D}, \boldsymbol{a}_I \vDash \varphi$ gilt;
  **local variables** $L$ : **array** $E_{\mathscr{A}}$ **of** $\mathscr{P}(\Phi(P, \boldsymbol{A}(\mathscr{D})))$;
**begin**
  **foreach** $e \in E_{\mathscr{A}}$ **do** $L(e) \leftarrow \varnothing$ **od**
  *check*($\varphi$);
  **output** $\varphi \in L(e_I)$
**end** DCTL.

Algorithmus IV.1. Ein Modelchecker für DCTL.

---

(a) $e_1 \xrightarrow[\mathscr{A},i]{a} e_2 \Rightarrow \forall \boldsymbol{x} \in \mathbf{CNF}_{\mathscr{A}}(e_1), \boldsymbol{y} \in \mathbf{CNF}_{\mathscr{A}}(e_2) \left( \boldsymbol{x} \xrightarrow[\mathscr{D},i]{a} \boldsymbol{y} \right)$,

(b) $\boldsymbol{x} \neq \boldsymbol{\epsilon}$ & $\boldsymbol{x} \xrightarrow[\mathscr{D},i]{a} \boldsymbol{y} \Rightarrow \exists e_1 \in \mathbf{AC}_{\mathscr{A},i}(\boldsymbol{x}), e_2 \in \mathbf{AC}_{\mathscr{A},i}(\boldsymbol{y}) \left( e_1 \xrightarrow[\mathscr{A},i]{a} e_2 \right)$,

(c) $\boldsymbol{\epsilon} \xrightarrow[\mathscr{D},i]{a} \boldsymbol{x} \Rightarrow \exists e \in E_{\mathscr{A}} \, (e \in \mathbf{AC}_{\mathscr{A},i}(\boldsymbol{x}))$.

Beweis. (a) Nehmen wir $e_1 \xrightarrow[\mathscr{A},i]{a} e_2$ für $e_1, e_2 \in E_{\mathscr{A}}$ an. Sei $\boldsymbol{x} \in \mathbf{CNF}_{\mathscr{A}}(e_1)$ und $\boldsymbol{y} \in \mathbf{CNF}_{\mathscr{A}}(e_2)$. Dann gilt nach den Lemmata III.1.6 und IV.1.4.(b) $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{CNF}_i(\mathscr{D})$: Lemma III.1.6 besagt, daß $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{LSSL}(\mathscr{D})$ sind, während Lemma IV.1.4.(b) $\boldsymbol{x}$ und $\boldsymbol{y}$ als $i$-lokale Konfigurationen ausweist. Leicht einzusehen ist, daß $\boldsymbol{x} < \boldsymbol{y}$ gilt. Mit Definition IV.3.1 jedoch folgt sofort $\boldsymbol{z} \notin \mathbf{CNF}_i(\mathscr{D})$ für alle $\boldsymbol{z}$ mit $\boldsymbol{x} < \boldsymbol{z} < \boldsymbol{y}$.

(b) Es gelte $\boldsymbol{x} \xrightarrow[\mathscr{D},i]{a} \boldsymbol{y}$. Sei $x \in \boldsymbol{x}$ und $y \in \boldsymbol{y}$, es gilt $x < y$. Setzen wir $e_1' =_{\text{Def}} \max(x)$ und $e_2' =_{\text{Def}} \max(y)$. Dann sind $\lambda_x(e_1') \in A_i$, $\lambda_x(e_2') = a \in A_i$, und für alle $e' \in E_y$ mit $H_x^y(e_1') <_y e' <_y e_2'$ gilt $e' \notin A_i$, da andernfalls das Semiwort $\boldsymbol{x}' =_{\text{Def}} \left[ y \left[ \leqslant_y^{-1}(e') \right] \right]$ in $\mathbf{CNF}_i(\mathscr{D})$ wäre, andererseits jedoch $\boldsymbol{x} < \boldsymbol{x}' < \boldsymbol{y}$ gelten würde. Da $\mathscr{A}$ vollständig bzgl. $\mathscr{D}$ ist, gibt es ein Semiwort $\boldsymbol{z} \in \mathbf{SL}(\mathscr{A})$, so daß $\boldsymbol{y} \leqslant \boldsymbol{z}$ gilt. Um die Ereignisse von $\boldsymbol{z}$ mit Ereignissen von $\mathscr{A}$ assoziieren zu können, wählen wir eine Semiordnung $z \in \boldsymbol{z}$ so, daß $E_z \subseteq E_{\mathscr{A}}$, und $<_z \subseteq <_{\mathscr{A}}$ gilt. Setzen wir nun $e_1 =_{\text{Def}} H_x^z(e_1')$ und $e_2 =_{\text{Def}} H_y^z(e_1')$; es gilt nun also $e_1 \in \mathbf{AC}_{\mathscr{A},i}(\boldsymbol{x})$ und $e_2 \in \mathbf{AC}_{\mathscr{A},i}(\boldsymbol{y})$, weiterhin ist $\lambda_{\mathscr{A}}(e_2) = a$.

Nehmen wir nun ein Ereignis $e \in E_{\mathscr{A}}$ mit $e_1 <_z e <_z e_2$ und $\lambda_{\mathscr{A}}(e) \in A_i$ an. Setzen wir $\boldsymbol{z}' =_{\text{Def}} [z \, [\leqslant_z^{-1}(e)]]$, so gilt $\boldsymbol{z}' \in \mathbf{CNF}_i(\mathscr{D})$ sowie $\boldsymbol{x} < \boldsymbol{z}' < \boldsymbol{y}$ im Widerspruch zur Voraussetzung $\boldsymbol{x} \xrightarrow[\mathscr{D},i]{a} \boldsymbol{y}$. Wir schließen $e_1 \xrightarrow[\mathscr{A},i]{a} e_2$.

---

**procedure** *check*$(\varphi_0 : \Phi(P, \boldsymbol{A}(\mathscr{D})))$ **is**
**begin**
  **case** $\varphi_0$ **of**
    $p[i]$: **foreach** $e \in E_{\mathscr{A}}$ **do if** $p \in \nu_i(e)$ **then** $L(e) \leftarrow L(e) \cup \{p[i]\}$ **fi od**
    $\neg\varphi$: *check*$(\varphi)$;
      **foreach** $e \in E_{\mathscr{A}}$ **do if** $\varphi \notin L(e)$ **then** $L(e) \leftarrow L(e) \cup \{\neg\varphi\}$ **fi od**
    $\varphi \vee \psi$: *check*$(\varphi)$; *check*$(\psi)$;
      **foreach** $e \in E_{\mathscr{A}}$ **do**
        **if** $\varphi \in L(e) \vee \psi \in L(e)$ **then** $L(e) \leftarrow L(e) \cup \{\varphi \vee \psi\}$ **fi**
      **od**
    $\langle a \rangle_i \varphi$: *check*$(\varphi)$;
      **foreach** $e \in E_{\mathscr{A}}$ **do**
        **if** $\exists e' \in E_{\mathscr{A}} \left( e \xrightarrow[\mathscr{A},i]{a} e' \;\&\; \varphi \in L(e') \right)$ **then** $L(e) \leftarrow L(e) \cup \{\langle a \rangle_i \varphi\}$ **fi**
      **od**
    $\boldsymbol{E}(\varphi\ \boldsymbol{U}_i\ \psi)$: *check*$(\varphi)$; *check*$(\psi)$;
      **foreach** $e \in E_{\mathscr{A}}$ **do** *check_eu*$(e, i, \varphi, \psi)$ **od**
    $\boldsymbol{A}(\varphi\ \boldsymbol{U}_i\ \psi)$: *check*$(\varphi)$; *check*$(\psi)$;
      **foreach** $e \in E_{\mathscr{A}}$ **do** *check_au*$(e, i, \varphi, \psi)$ **od**
  **esac**
**end** *check*.

ALGORITHMUS IV.2. Prozedur *check*

(c) kann durch eine Argumentation ähnlich zu der für Fall (b) nachgewiesen werden; wir verzichten darauf, diese explizit aufzuführen. □

Damit können wir einen Modelchecker für DCTL bestimmen. Definieren wir die *Fischer-Ladner-Hülle* oder kurz die *Hülle über einer DCTL-Formel* über einer Formel $\varphi_0 \in \Phi(P, \boldsymbol{A}(\mathscr{D}))$ als die kleinste Menge $C(\varphi_0)$, die die folgenden Elemente enthält:

(a) $\varphi_0 \in C(\varphi_0)$.
(b) Sind $\neg\varphi$ oder $\langle a \rangle_i \varphi$ in $C(\varphi_0)$ enthalten, so ist auch $\varphi \in C(\varphi_0)$.
(c) Sind $\varphi \vee \psi$, $\boldsymbol{E}(\varphi\ \boldsymbol{U}_i\ \psi)$ oder $\boldsymbol{A}(\varphi\ \boldsymbol{U}_i\ \psi)$ in $C(\varphi_0)$ enthalten, so sind auch $\varphi, \psi \in C(\varphi_0)$.

Algorithmus IV.1 berechnet für die Eingabeformel $\varphi_0$ eine Beschriftung $L : E_{\mathscr{A}} \to \mathscr{P}(\Phi(P, \boldsymbol{A}(\mathscr{D})))$, so daß $L(e) \subseteq C(\varphi_0)$ mit

$$\varphi \in L(e) \Leftrightarrow \forall \boldsymbol{x} \in \mathbf{CNF}_{\mathscr{A}}(e)\,(\mathscr{D}, \boldsymbol{x} \vDash \varphi)$$

gilt.

**procedure** *check_eu*(**in** $e : E_{\mathscr{A}}$; **in** $i : J$; **in** $\varphi, \psi : \Phi(P, \boldsymbol{A}(\mathscr{D}))$) **is**
  **local variables** $b : \mathbb{B}$;
**begin**
  **if** $\boldsymbol{E}\,(\varphi\ \boldsymbol{U}_i\ \psi) \notin L(e)$ **then**
    **if** $\psi \in L(e)$ **then**
      $L(e) \leftarrow L(e) \cup \{\boldsymbol{E}\,(\varphi\ \boldsymbol{U}_i\ \psi)\}$
    **elsif** $\varphi \in L(e)$ **then**
      $b \leftarrow$ false;
      **foreach** $e' \in E_{\mathscr{A}} : e \xrightarrow[\mathscr{A}]{i} e'$ **do**
        *check_eu*$(e', i, \varphi, \psi)$; $b \leftarrow b \vee \boldsymbol{E}\,(\varphi\ \boldsymbol{U}_i\ \psi) \in L(e')$;
      **od**;
      **if** $b$ **then** $L(e) \leftarrow L(e) \cup \{\boldsymbol{E}\,(\varphi\ \boldsymbol{U}_i\ \psi)\}$ **fi**
    **fi**
  **fi**
**end** *check_au*

ALGORITHMUS IV.3. Berechnung von $L(e)$ für $\boldsymbol{E}\,(\varphi\ \boldsymbol{U}_i\ \psi)$.

---

Wie bestimmen wir nun, ob $\mathscr{D}, \boldsymbol{\epsilon} \vDash \varphi_0$ gilt? Für das leere Semiwort $\boldsymbol{\epsilon}$ gibt es kein Ereignis $e \in E_{\mathscr{A}}$, für das die Beziehung

$$\varphi \in L(e) \Leftrightarrow \mathscr{D}, \boldsymbol{\epsilon} \vDash \varphi$$

gelten könnte. Erinnern wir uns daran, daß Spursysteme $\mathscr{T}$, die als Eingaben für einen der beiden Algorithmen III.1 oder III.3 zur Prozeßautomatenerzeugung in Frage kommen, als initialisiert angenommen waren. Wir gehen nun der Einfachheit halber davon aus, daß $\mathscr{A}$ ein Prozeßautomat zu einem durch eine Aktion $a_I$ initialisierten Spursystem $\mathscr{T}$ ist, wobei $a_I$ nachträglich zur algorithmischen Behandlung von $\mathscr{T}$ hinzugefügt wurde: Ist $\mathscr{D}$ ein verteiltes System, nehmen wir also an, daß die Prozeßautomatenerzeugung und das anschließende Modelchecking für das verteilte System $\mathscr{D}^*$ mit $\mathscr{T}_{\mathscr{D}^*} = \mathscr{T}^*_{\mathscr{D}}$ durchgeführt wird (vgl. Def I.3.53). Ist $e_I \in \mathscr{A}$ das eindeutig bestimmte Ereignis mit $\lambda_{\mathscr{A}}(e_I) = a_I$, so gilt offenbar

$$\mathscr{D}, \boldsymbol{\epsilon} \vDash \varphi \Leftrightarrow \mathscr{D}^*, \boldsymbol{a_I} \vDash \varphi \Leftrightarrow \varphi \in L(e_I).$$

Die Prozedur *check* (Algorithmus IV.2) führt eine „Syntaxanalyse“ der Eingabeformel $\varphi_0$ durch und bestimmt entsprechend der syntaktischen Struktur von $\varphi_0$ für alle $e \in E_{\mathscr{A}}$, wie ausgehend von den bereits betrachteten Teilformeln bestimmt werden kann, ob $\varphi_0$ in $L(e)$ eingefügt werden muß. Für die Fälle $p[i]$, $\neg\varphi$, $\varphi \vee \psi$ und $\langle a \rangle_i\, \varphi$ wird dies in der offensichtlichen Weise bestimmt, für die Fälle $\boldsymbol{E}\,(\varphi\ \boldsymbol{U}_i\ \psi)$ und $\boldsymbol{A}\,(\varphi\ \boldsymbol{U}_i\ \psi)$ werden die Prozeduren *check_eu* und *check_au* aufgerufen (Algorithmen IV.3 und IV.4), die für alle von $e$ aus über $i$-Schritte erreichbaren Ereignisse $e'$ die Auswertung der $\boldsymbol{U}_i$-Operatoren entsprechend der in Lemma IV.1.19 genannten Äquivalenzen vornehmen.

---

**procedure** *check_au*(**in** $e : E_{\mathscr{A}}$; **in** $i : J$; **in** $\varphi, \psi : \Phi(P, \boldsymbol{A}(\mathscr{D}))$) **is**
  **local variables** $b : \mathbb{B}$;
**begin**
  **if** $\boldsymbol{A}\,(\varphi\ \boldsymbol{U}_i\ \psi) \notin L(e)$ **then**
    **if** $\psi \in L(e)$ **then**
      $L(e) \leftarrow L(e) \cup \{\boldsymbol{A}\,(\varphi\ \boldsymbol{U}_i\ \psi)\}$
    **elsif** $\varphi \in L(e)$ **then**
      $b \leftarrow$ true;
      **foreach** $e' \in E_{\mathscr{A}} : e \xrightarrow[\mathscr{A}]{i} e'$ **do**
        *check_au*$(e', i, \varphi, \psi)$; $b \leftarrow b$ & $\boldsymbol{A}\,(\varphi\ \boldsymbol{U}_i\ \psi) \in L(e')$;
      **od**;
      **if** $b$ **then** $L(e) \leftarrow L(e) \cup \{\boldsymbol{A}\,(\varphi\ \boldsymbol{U}_i\ \psi)\}$ **fi**
    **fi**
  **fi**
**end** *check_au*

ALGORITHMUS IV.4. Berechnung von $L(e)$ für $\boldsymbol{A}\,(\varphi\ \boldsymbol{U}_i\ \psi)$.

Die beiden folgenden Theoreme sind offensichtlich; wir verzichten auf ausführliche Beweise.

IV.3.5. THEOREM. *Ist $\mathscr{A}$ ein Prozeßautomat zu dem initialisierten verteilten System $\mathscr{D}^*$, $\varphi$ eine DCTL-Formel und $\{\nu_i\}_{i\in J}$ eine Familie von Wertzuweisungen, so hat Algorithmus IV.1 für diese Eingaben den Rückgabewert* true *gdw. $\mathscr{D} \models \varphi$ vorliegt.*

IV.3.6. THEOREM. *Sei $\mathscr{A}$ ein Prozeßautomat zu dem initialisierten verteilten System $\mathscr{D}^*$ $\varphi$ eine DCTL-Formel und $\{\nu_i\}_{i\in J}$ eine Familie von Wertzuweisungen. Die Zeitkomplexität von Algorithmus IV.1 ist für solche Eingaben von der Ordnung $O\,(|C(\varphi)| \cdot |E_{\mathscr{A}}|^2)$.*

BEWEIS. Der zweite Faktor, $|E_{\mathscr{A}}|^2$ ist eine obere Schranke für die maximale Anzahl der Paare $e_1, e_2 \in E_{\mathscr{A}}$ mit $e_1 \xrightarrow[\mathscr{A}]{i} e_2$. □

# Ausblick

Dieser letzte Teil soll einen Ausblick auf verschiedene Aspekte liefern, die in dieser Arbeit nicht betrachtet werden konnten. Einerseits stellt sich die Frage, ob neben dem Modelchecking für DCTL andere Anwendungsgebiete für Prozeßautomaten existieren.

Ulrich [**87**] verwendet Prozeßautomaten zur Ableitung von Testfällen für Kommunikationsprotokolle. In diesem Anwendungsgebiet ist jedoch die Verwendung von Timern von zentraler Bedeutung. Deshalb stellt sich die Frage, ob und wie Prozeßautomaten für Formalismen definiert und erzeugt werden können, denen ein explizites Zeit- oder Timermodell zugrundeliegt.

Eine weitere interessante Problemstellung ergibt sich aus der Frage, ob und wie ein prozeßautomatenbasierter Modelchecker für eine temporale Logik, der ein lineares Zeitmodell zugrunde liegt, gefunden werden kann. Ein Ansatz, der analog zu den in den Arbeiten [**18, 96**] beschriebenen Verfahren verläuft, scheint hier aussichtsreich zu sein. Darüberhinaus kann natürlich überlegt werden, ob weitere temporale Operatoren zu DCTL hinzugefügt werden können, z .B. Fixpunktoperatoren oder Operatoren, die sich auf die Vergangenheit einer lokalen Konfiguration beziehen.

Wie wir gesehen haben, ist die Theorie der (unstrukturierten) Petri-Netze an vielen Stellen dieser Arbeit inspirierend gewesen, insbesondere (!) im Zusammenhang mit der Herleitung und Verwendung von Systemkomponenten. Aber natürlich gibt es eine Vielzahl anderer Formalismen zur Beschreibung verteilter Systeme. Es ist also zu erwarten, daß Ideen und Techniken, die sich auf solche Beschreibungsmittel beziehen, ebenfalls nützliche Hinweise auf Verbesserungen unseres Ansatzes liefern können.

**Eine praktischer Testfall.** Eine Implementierung des in Abschnitt IV.3 vorgestellen Modelcheckers konnte in dem zur Verfügung stehenden Zeitrahmen nicht mehr vorgenommen werden, eine prototypische Implementierung der Algorithmen zur Erzeugung von Prozeßautomaten für sichere Petri-Netze zusammen mit einer Agentenverteilung existiert jedoch. Die in Abb. IV.10 gezeigte Tabelle stellt einige Ergebnisse für das Beispiel aus [**38**] zusammen. Dabei handelt es sich um ein Petri-Netz-Modell einer Steuerung einer Produktionszelle. Diese Zelle umfaßt in der *offenen Version* fünf physikalische Komponenten: Zwei Fließbänder, einen Hubdrehtisch, eine Presse und einen zweiarmigen Roboter. In der *geschlossenen Version* ist ein Transportkran hinzugefügt, der verarbeitete Werkstücke in den

| Beispiel | P/T | $\Theta$ | $\lvert S_{\mathscr{D}}\rvert$ | Auffaltung | Prozeßautomat | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | (B/E) | Zeit | $\lvert S_{\mathscr{A}}\rvert$ | $\lvert X_{\mathscr{A}}\rvert$ | $\lvert E_{\mathscr{A}}\rvert$ |
| Geschl. System | 232/203 | 45 | | | | | | |
| mit 1 WS | | | 30,952 | 690/316 | 24 | 3 | 3 | 225 |
| mit 2 WS | | | 543,480 | 2773/1348 | 39 | 22 | 35 | 575 |
| mit 3 WS | | | 2,941,944 | 2009/960 | 37 | 60 | 98 | 909 |
| mit 4 WS | | | 7,061,927 | 2164/1035 | 28 | 78 | 128 | 1053 |
| mit 5 WS | | | 1,657,242 | 1670/792 | 11 | 30 | 48 | 686 |
| Offenes System | 199/177 | 41 | 2,706,983 | 2773/1348 | 7 | 15 | 21 | 503 |

ABBILDUNG IV.10. Evaluation von Algorithmus III.3.

Produktionsprozeß zurückführt; damit wird das System unabhängig von weiteren Komponenten. In diesem geschlossenen System können bis zu fünf Werkstücke zirkulieren. Wir haben das Petri-Netz-Modell aus [**38**] mit einer Agentenverteilung versehen, um ein verteiltes System zu erhalten.

Die Tabelle in Abb. IV.10 enthält die folgenden Daten: P/T gibt die Anzahl der Plätze und Transitionen des Netzmodells an, $\Theta$ ist die Anzahl der zugewiesenen Komponenten. $|S_{\mathscr{D}}|$ ist die Anzahl der Zustände des von dem jeweiligen Netzsystem induzierten Spursystems. Die Größe des erzeugten Prozeßautomaten $\mathscr{A}$ wird durch die Anzahl der globalen Zustände $|S_{\mathscr{A}}|$, die Anzahl der Semiordnungen $|X_{\mathscr{A}}|$ und die Anzahl der Ereignisse $|E_{\mathscr{A}}|$ angegeben. Die Erzeugungszeiten sind in Sekunden angegeben, die Messungen wurden auf einer SPARC Station 20 mit 64 MB Hauptspeicher vorgenommen.

Gegenübergestellt werden die Zahlen der Größe der Auffaltungen der Netzsysteme. Dabei gibt B/E die Anzahl der Bedingungen/Ereignisse der Auffaltung an. Die Anzahlen der Ereignisse der Auffaltung und der des Prozeßautomaten ist eine gute Vergleichsgröße.

**Implementierung.** Als zentrale Datenstruktur der prototypischen Implementierung wurde eine generische Version von AVL-Bäumen verwendet; Instanziierungen mit verschiedenen Datentypen liefern Mengen, Abbildungen, Relationen, Graphen usw. Obwohl sich diese Vorgehensweise als äußerst flexibel, fehlerunanfällig und intuitiv erweisen hat, hätten an verschiedenen Stellen natürlich bessere Möglichkeiten zur Datenhaltung gefunden werden können (Reihungen, Hashtabellen usw.).

Für Semiordnungen $x$ wird natürlich nicht die vollständige Relation $<_x$, sondern lediglich $\lessdot_x$ im Speicher gehalten. Die Implementierung der $\circ_\Sigma$-Operation kann dann als eine Variante des topologischen Sortierens erfolgen. Zur Implementierung des DCTL-Modelcheckers ist es allerdings sinnvoller, die Relation $\xrightarrow[\mathscr{A}]{i}$ für Agenten $i$ zu repräsentieren, tatsächlich ist die algorithmische Behandlung dieser Relation sogar einfacher als die der $\lessdot_x$-Relation.

Die größte Schwierigkeit bei der Implementierung des Modelcheckers für DCTL liegt offenbar in der Berechnung und Repräsentation der Schrittrelation $\xrightarrow[\mathcal{A},i]{a}$ (vgl. Def. IV.3.1). Für solche Ereignisse $e_1$ und $e_2$ in einem Prozeßautomaten $\mathcal{A}$, für die $x_{\mathcal{A}}(e_1) = x = x_{\mathcal{A}}(e_2)$ gilt, kann die Beziehung $e_1 \xrightarrow[\mathcal{A},i]{a} e_2$ schon bei der Erzeugung des Prozeßautomaten überprüft werden. Gilt andererseits $x_{\mathcal{A}}(e_1) \neq x_{\mathcal{A}}(e_2)$, so können die maximalen Ereignisse in $x_{\mathcal{A}}(e_1)$, die mit Aktionen aus $A_i$ beschriftet sind, ebenso wie die mit Aktionen aus $A_i$ beschrifteten minimalen Ereignisse aus $x_{\mathcal{A}}(e_2)$ bei der Prozeßautomatenerzeugung bestimmt und abgespeichert werden (tatsächlich ist diese Vorgehensweise bereits nützlich bei der Implementierung der $\circ_\Sigma$-Operation). Die Berechnung einer $\xrightarrow[\mathcal{A},i]{a}$-Beziehung kann dann entweder vor der Überprüfung der Gültigkeit einer Eingabeformel oder aber währenddessen stattfinden (welche Möglichkeit in welchen Fällen sinnvoller ist, muß experimentell bestimmt werden). Es ist also während des Modelchecking-Prozesses unnötig, zur Bestimmung der Schrittrelation nocheinmal eine der Semiordnungen in $X_{\mathcal{A}}$ zu traversieren.

**Zusammenfassung.** Zum Abschluß wollen wir nocheinmal die wesentlichen Ergebnisse dieser Arbeit nennen:

Ist $\Sigma$ ein Spuralphabet, so stellen Semiwörter $\boldsymbol{x} \in \mathbf{LSW}(\Sigma)$ eine adäquate mathematische Struktur zur Erklärung der Semantik von Spursystemen über $\Sigma$ dar, für die eine reichhaltige Theorie gebildet werden kann. Nicht zuletzt ist durch die Verwendung der Operationen tr und ord ein Rückgriff auf die Theorie der Mazurkiewiczspuren möglich.

Jedes Spursystem kann als nichttriviales verteiltes System aufgefaßt werden. Das befreit die Annahme einer Komponentenstruktur von ihrem *ad-hoc*-Charakter.

Prozeßautomaten stellen einen alternativen Ansatz zur halbordnungsbasierten Repräsentation von Systemverhalten dar. Es existiert ein wohldefinierter Sprachbegriff. Weiterhin existieren effiziente Erzeugungsalgorithmen für Prozeßautomaten.

Verteilte agentenbasierte Logiken erlauben die Formulierung einer Reihe interessanter Eigenschaften, die in den klassischen, auf einer sequentiellen Semantik beruhenden temporalen Logiken wegen der Unmöglichkeit, Verzahnungen nebenläufiger Ereignisse von durch Kausalbeziehungen bestimmten Abfolgen zu unterscheiden. Für die in dieser Arbeit behandelte Logik existiert ein Modelchecking-Algorithmus, der auf Prozeßautomaten operiert.

# Symbolverzeichnis

## Algorithmen



## Allgemeines

### DCTL allgemein



### DCTL-Modelchecking



### DCTL-Operatoren

### Halb- und Semiwörter



### Invarianten

### Kommunizierende seq. Prozesse



### Netzsysteme



### Positionszuweisungen



### Prozeßautomaten



### Regionen



### Registersysteme

### Spuralphabete



### Spuren



### Systemspezifikation mit DCTL



### Transitions- und Spursysteme



### Verteilte Systeme

# Stichwortverzeichnis

Fettgedruckte Zahlen geben die Seite an, auf der sich die Definition eines Begriffs befindet. Schräggestellte Seitenangaben bedeuten, daß ein Begriff hier primär diskutiert wird.

**L**



**M**



**N**

# Literaturverzeichnis


[1] Abramsky, S. und A. Jung: *Domain Theory*. In: Abramsky, S., D. M. Gabbay und T. S. E. Maibaum (Hrsg.): *Handbook of Logic in Computer Science*, Bd. 2 — Semantic Structures, Kap. 1, S. 1 – 168. Clarendon Prss, Oxford, 1994.

[2] Badouel, E. und P. Darondeau: *Trace Nets and Process Automata*. Acta Informatica, 32:647 – 679, 1995.

[3] Badouel, E. und P. Darondeau: *Theory of Regions*. In: Reisig, W. und G. Rozenberg (Hrsg.): *Lectures on Petri Nets I: Basic Modells*, Nr. 1491 in *Lecture Notes in Computer Science*, S. 529 – 586. Springer-Verlag, 1998. Lecture Notes of the 3rd Advanced Course on Petri Nets, Dagstuhl (1996).

[4] Barnes, J. (Hrsg.): *Ada 95 Rationale: The Language, The Standard Libraries*. Nr. 1247 in *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[5] Ben-Ari, M., Z. Manna und A. Pnueli: *The Temporal Logic of Branching Time*. In: *Proc. of the 8th Annual ACM Symposium on Principles of Programming Languages*, S. 164 – 176, New York, 1981.

[6] Bernardinello, L.: *Synthesis of Net Systems*. In: *Proc. of the Int. Conf. on Application and Theory of Petri Nets (ICATPN'93)*, Nr. 691 in *Lecture Notes in Computer Science*, S. 89 – 105. Springer-Verlag, 1993.

[7] Best, E., R. Devillers und M. Koutny: *Petri Nets, Process Algebras and Programming Languages*. In: Reisig, W. und G. Rozenberg (Hrsg.): *Lectures on Petri Nets II: Applications*, Nr. 1492 in *Lecture Notes in Computer Science*, S. 1–84. Springer-Verlag, 1998. Lecture Notes of the 3rd Advanced Course on Petri Nets, Dagstuhl (1996).

[8] Best, E., R. Devillers und M. Koutny: *Petri Net Algebra*. EATCS Monographs on Theoretical Computer Science Series. Springer-Verlag, 2001.

[9] Best, E. und C. Fernández: *Nonsequential Processes*, Bd. 13 d. Reihe *EATCS Monographs on Theoretical Computer Science Series*. Springer-Verlag, 1988.

[10] Birkhoff, G.: *Lattice Theory*, Bd. 25 d. Reihe *Colloquium Publications*. American Mathematical Society, Providence, Rhode Island, 3 Aufl., 1967.

[11] Booch, G., J. Rumbaugh und I. Jacobson: *The Unified Modeling Language User Guide*. Addison-Wesley, 1998.

[12] Bron, C. und J. Kerbosch: *Algorithm 457, Finding all Cliques of an Undirected Graph*. Communications of the ACM, 16:575–577, 1973.

[13] Bryant, E. R.: *Symbolic Boolean Manipulation with Ordered Binary-Dicision Diagrams*. ACM Computing Survey, 24(3):293–318, 1992.

[14] Bryant, R. E.: *Graph-Based Algorithms for Boolean Function Manipulation*. IEEE Transactions on Computers, C-35(6):677 – 691, 1986.

[15] Burch, J. R., E. M. Clarke und K. L. McMillan: *Symbolic Model Checking:* $10^{20}$ *States and Beyond*. In: *The 5th Annual IEEE Symposium on Logic in Computer Science*, S. 428 – 439, New York, 1990. IEEE.

[16] Clarke, E. M., E. A. Emerson und A. P. Sistla: *Automatic Verification of Finite-State Concurrenct Systems Using Temporal Logic Specifications*. ACM Transactions onf Programming Languages and Systems, 8(2):244 – 263, 1986.

[17] Courcoubetis, C., M. Vardi, P. Wolper und M. Yannakakis: *Memory-Efficient Algorithms for the Verification of Temporal Properties*. Formal Methods in System Design, 1:275 – 288, 1992.

[18] Couvreur, J.-M. und D. Poitrenaud: *Model Checking based on Occurence Net Graph*. In: *Proc. of Formal Description Techniques IX, Theory, Application, and Tools*, S. 380 – 295, 1996.

[19] Desel, J.: *Basic Linear Algebraic Techniques for Place/Transition Nets*. In: Reisig, W. und G. Rozenberg (Hrsg.): *Lectures on Petri Nets I: Basic Modells*, Nr. 1491 in *Lecture Notes in Computer Science*, S. 257 – 308. Springer-Verlag, 1998. Lecture Notes of the 3rd Advanced Course on Petri Nets, Dagstuhl (1996).

[20] Desel, J.: *Petrinetze, lineare Algebra und lineare Programmierung*, Bd. 26 d. Reihe *Teubner-Texte zur Informatik*. G. B. Teubner, Stuttgart – Leipzig, 1998.

[21] Desel, J. und W. Reisig: *The Synthesis Problem for Petri Nets*. Acta Informatica, 33:297 – 315, 1995.

[22] Deussen, P.: *Algorithmic Aspects of Concurrent Automata*. In: Burkhard, H.-D., L. Czaja und P. Starke (Hrsg.): *Workshop on Concurrency, Specification & Programming '98*, Nr. 110 in *Informatik-Berichte*, S. 39–50, Berlin, 1998. Humboldt Univ. zu Berlin.

[23] Deussen, P.: *Concurrent Automata*. Techn. Ber. 1-05/1998, Brandenburg Tech. Univ. Cottbus, 1998.

[24] Deussen, P.: *Improvements of Concurrent Automata Generation*. Techn. Ber. I-08/1998, Brandenburg Tech. Univ. Cottbus, 1999.

[25] Deussen, P.: *Partial Order Verification of Programmable Logic Controllers*. In: Colom, J. S. und M. Koutny (Hrsg.): *Appications and Theory of Petri Nets 2001*, Nr. 2075 in *Lecture Notes in Computer Science*, S. 144 – 163. Springer-Verlag, 2001.

[26] Droste, M.: *Concurrency, Automata and Domains*. In: Paterson, M. S. (Hrsg.): *Automata, Languages and Programming*, Nr. 443 in *Lecture Notes in Computer Science*, S. 195 – 202. Springer-Verlag, 1990.

[27] Ehrenfeucht, A. und G. Rozenberg: *Partial 2-Structures; Part II: State Space of Concurrent Systems:*. Acta Informatica, 27:348 – 368, 1990.

[28] Emerson, E. A.: *Temporal and Model Logic*. In: Leeuwen, J. van (Hrsg.): *Handbook of Theoretical Computer Science*, Kap. 16, S. 997 – 1072. Elsevier Science Publishers B.V., 1990.

[29] Engelfriet, J.: *Branching processes of Petri nets*. Acta Inf., 25:575–591, 1991.

[30] Esparza, J.: *Model checking using net unfoldings*. Science of Computer Programming, 23:151–195, 1994.

[31] Esparza, J., S. Römer und W. Vogler: *An Improvement of McMillan's Unfolding Algorithm*. Techn. Ber. SFB-Report 342/12/95 A, Tech. Univ. of München, 1995.

[32] Gerth, R., D. Peled, M. Vardi und P. Wolper: *Simple On-the-fly Automatic Verification of Linear Temporal Logik*. In: *Proc. of 15th Int. Symposium on Protocol Specification, Testing and Verification (PSTV'95)*, S. 3 – 18, 1995.

[33] Godefroid, P.: *Partial-Order Methods for the Verification of Concurrent Systems*. Nr. 1032 in *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[34] Goossens, M., F. Mittelbach und A. Samarin: *The LaTeX Companion*. Addison-Wesley, Reading, 1993.

[35] Grabowski, J.: *On partial languages*. Fund. Inform, 4(2):427–498, 1981.

[36] GUNTER, C. A. und D. S. SCOTT: *Semantic Domains*. In: LEEUWEN, J. VAN (Hrsg.): *Handbook of Theoretical Computer Science*, Kap. 12, S. 633 – g74. Elsevier Science Publishers B.V., 1990.

[37] HEINER, M.: *Verification and Optimization of Control Programs by Petri Nets without State Explosion*. Proc. 2nd Int. Workshop on Manufacturing and Petri Nets held at Int. Conf. on Application and Theory of Petri Nets, S. 69–84, 1997.

[38] HEINER, M., P. DEUSSEN und S. SPRANGER: *A Case Study in Design and Verification of Manufacturing System Control Software with Hierarchical Petri Nets*. The Int. Journal of Advanced Manufacturing Technology, 15:139–152, 1999.

[39] HOARE, C. A. R.: *Communicating Sequential Processes*. Prentice Hall, 1998.

[40] HUHN, M., P. NIEBERT und F. WALLNER: *Verification Based on Local States*. In: *Proc. of TACAS'98*, Nr. 1384 in *Lecture Notes in Computer Science*, S. 36 – 51. Springer-Verlag, 1998.

[41] HUHN, M., P. NIEBERT und F. WALLNER: *Model Checking Logics for Communicating Sequential Agents*. In: *Proc. of FOSSACS'99*, Bd. 1578 d. Reihe *Lectuce Notes in Computer Science*, S. 227 – 242. Springer-Verlag, 1999.

[42] HULGAARD, H.: *Timing Analysis and Verification of Timed Asynchronous Circuits*. Doktorarbeit, Univ. of Washington, 1995.

[43] HULGAARD, H. und S. M. BURNS: *Bounded Delay Timing Analysis of a Class of CSP Programs*. Formal Methods in System Design, 11:265 – 294, 1997.

[44] HULGAARD, H., S. M. BURNS, T. AMON und G. BORRIELLO: *An Algorithm for Exact Bounds on the Time Separation of Events in Concurrent Systems*. IEEE Transactions on Computers, 44(11):1306 –1317, 1995.

[45] KINDLER, E.: *Modularer Entwurf verteilter Systeme mit Petrinetzen*. Dieter Belz Verlag, Berlin, 1995.

[46] KNUTH, D. E.: *The* METAFONT*Book*, Bd. C d. Reihe *Computers and Typesetting*. Addison-Wesley, Reading, 1986.

[47] KNUTH, D. E.: *The TEXBook*, Bd. A d. Reihe *Computers and Typesetting*. Addison-Wesley, Reading, 1986.

[48] LOECKX, J. und K. SIEBER: *The Foundation of Program Verification*. B. G. Teubner, Stuttgart, 2 Aufl., 1987.

[49] MAUW, S. und M. A. RENIERS: *Operational Semantics for MSC'96*. In: CAVALLI, A. und D. VINCENT (Hrsg.): *Tutorials of the Eighth SDL Forum SDl'97: Time for Testing - SDL, MSC and Trends*, S. 135–152, Evry, France, 1997. Institut national des télécommunications.

[50] MAZURKIEWICZ, A.: *Concurrent program schemes ant their interpretations*. Techn. Ber. DAIMI PB. 78, Aarhus University, Aarhus, 1977.

[51] MAZURKIEWICZ, A.: *Introduction to Trace Theory*. In: DIEKERT, V. und G. ROZENBERG (Hrsg.): *The Book of Traces*, Kap. 1, S. 3 – 42. World Scientific, Singapore — New Jersey — London — Hong Kong, 1995.

[52] MCMILLAN, K. L.: *Symbolic Model Checking: An Approach to the State Explosion Problem*. Doktorarbeit, School of Computer Science, Carnegie-Mellon Univ., 1992.

[53] MCMILLAN, K. L.: *Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits*. In: *Proc. of the 4th Workshop on Computer Aided Verification*, S. 164–174, Montreal, 1992.

[54] MEHLHORN, K.: *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1984.

[55] MELZER, S.: *Verification of Distibuted Systems Using Linear and Constraint Programming*. Doktorarbeit, Techn. Univ. München, 1998.

[56] MELZER, S. und J. ESPARZA: *Verification of System Properties via Integer Programming*. In: NIELSON, H. R. (Hrsg.): *Programming Languages and Systems—ESOP'96*, Nr. 1058 in *Lecture Notes in Computer Science*, S. 250–264. Springer-Verlag, 1996.

[57] MILNER, R.: *Communication and Concurrency*. Prentice Hall, 1989.

[58] MURATA, T.: *Petri Nets: Properties, Analysis, and Applications*. Proc. of the IEEE, 77(4):541 – 580, 1989.

[59] NIELSEN, M., G. PLOTKIN und G. WINSKEL: *Petri Nets, Event Structures and Domains, Part I*. Theoretical Computer Science, 13:85–108, 1981.

[60] NIELSEN, M., G. ROZENBERG und P. S. THIAGARAJAN: *Behavioural Notions for Elementary Net Systems*. Distributed Computing, 4:45 – 57, 1990.

[61] NIELSEN, M., G. ROZENBERG und P. S. THIAGARAJAN: *Elementary Transition Systems*. Theoretical Computer Science, 96:3 – 33, 1992.

[62] NIELSEN, M. und G. WINSKEL: *Models of Concurrency*. In: ABRAMSKY, S., D. M. GABBAY und T. S. E. MAIBAUM (Hrsg.): *Handbook of Logic in Computer Science*, Bd. 4 — Semantic Modelling, Kap. 1, S. 1 – 148. Clarendon Press, Oxford, 1995.

[63] PASTOR, E., O. ROIG, J. CORTADELLA und R. M. BADIA: *Petri Net Analysis Using Boolean Manipulation*. In: VELETTE, R. (Hrsg.): *Proc. of the 15th Int. Conf. on Application and Theory of Petri Nets*, Nr. 815 in *Lecture Notes in Computer Science*, S. 416 – 435. Springer-Verlag, 1994.

[64] *PED — A Petri Net Editor*. http://www.btu-cottbus.de/wwwdssz.

[65] PENZEK, W. und R. KUIPER: *Traces and Logic*. In: DIEKERT, V. und G. ROZENBERG (Hrsg.): *The Book of Traces*, Kap. 1, S. 307 – 390. World Scientific, Singapore — New Jersey — London — Hong Kong, 1995.

[66] PETERSON, J. L.: *Petri Net Theoey and the Modelling of Systems*. Prentice-Hall, Inc., Engelwood Cliffs, N.J., 1981.

[67] PRATT, V.: *Modelling Concurrency with partial orders*. International Journal of Parallel Programming, 15(1):33–71, 1986.

[68] PRATT, V.: *Debate'90: An Electronic Discussion on True Concurrency*. In: PELED, D. A., V. PRATT und G. J. HOLZMANN (Hrsg.): *Proc. DIMACS Workshop on Partial Order Methods in Verification*, Bd. 29 d. Reihe *DIMACS—Series on Discrete Mathematics and Theoretical Computer Science*, S. 359–403. American Mathematical Society, 1996.

[69] PROBST, D. K. und H. F. LI: *Modelling Reactive Processes Using Partial Orders*. In: *Semantics for Concurrency*, Workshops in Coputing, S. 324 – 343, Leicester, 1990. Springer-Verlag.

[70] PROBST, D. K. und H. F. LI: *Partial-Order Model Checking: A Guide for the Perplexed*. In: *Proc. of CAV'91*, Nr. 575 in *Lecture Notes in Computer Science*, S. 322 – 331. Springer-Verlag, 1991.

[71] RAMANUJAM, R.: *Locally Linear Time Temporal Logic*. In: *Proc. of the 11th Annual IEEE Symposium on Logic in Computer Science*, S. 118 – 127. IEEE Computer Society Press, 1996.

[72] REINGOLD, E. M., J. NIEVERGELT und N. DEO: *Combinatorial Algorithms, Theory and Practice*. Prentice Hall, Englewood Cliffs, N. J., 1977.

[73] REISIG, W.: *Petrinetze: Eine Einführung*. Studienreihe Informatik. Springer-Verlag, 1992.

[74] ROZENBERG, G.: *Behaviour of Elementary Net Systems*. In: BRAUER, W. (Hrsg.): *Petri Nets: Central Models and their Properties; Advances in Petri Nets; Proc. of an Advanced Curse, Vol. 1*, Nr. 254 in *Lecture Notes in Computer Science*, S. 60–94, Berlin-Heidelberg-New York, 1986. Springer-Verlag.

[75] ROZENBERG, G. und J. ENGELFRIET: *Elementary net systems*. In: REISIG, W. und G. ROZENBERG (Hrsg.): *Lectures on Petri Nets I: Basic Modells*, Nr. 1491 in *Lecture Notes in*

*Computer Science*, S. 12 – 121. Springer-Verlag, 1998. Lecture Notes of the 3rd Advanced Course on Petri Nets, Dagstuhl (1996).

[76] Silva, M., E. Teruel und J. M. Colom: *Linear Algebraic and Linear Programming Techniques for the Analysis of Place/Transition Net Systems*. In: Reisig, W. und G. Rozenberg (Hrsg.): *Lectures on Petri Nets I: Basic Modells*, Nr. 1491 in *Lecture Notes in Computer Science*, S. 309 – 373. Springer-Verlag, 1998. Lecture Notes of the 3rd Advanced Course on Petri Nets, Dagstuhl (1996).

[77] Spranger, J.: *Symbolische LTL-Verifikation von Petrinetzen*. Doktorarbeit, Brandenburgische Technische Univ. Cottbus, 2001.
verfügbar unter `www.ub.tu-cottbus.de/hss/diss/fak1/spranger_j`.

[78] Starke, P. H.: *Processes in Petri nets*. J. Inf. Process. Cybern. EIK, 17(8/9):389–416, 1981.

[79] Starke, P. H.: *Graph Grammars for Petri Net Processes*. J. Inf. Process. Cybern. EIK, 19(4/5):199–233, 1983.

[80] Starke, P. H.: *Multiprocessor Systems and Their Concurrency*. J. Inf. Process. Cybern. EIK, 20(4):207–427, 1984.

[81] Starke, P. H.: *Analyse von Petri-Netz-Modellen*. G. B. Teubner, Stuttgart, 1990.

[82] Stirling, C.: *Modal and Temporal Logics*. In: Abramsky, S., D. M. Gabbay und T. S. E. Maibaum (Hrsg.): *Handbook of Logic in Computer Science*, Bd. 2: Background: Computational Structures, S. 477 – 563, Oxford, 1992. Clarendon Press.

[83] Taft, S. T. und R. A. Duff (Hrsg.): *Ada 95 Reference Manual: Language and Standard Libraries, International Standard ISO/IEC 8652:1995(E)*. Nr. 1246 in *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[84] Thiagarajan, P. S.: *A Trace Based Extension of Linear Time Temporal Logic*. In: *Proc. of the 9th Annual IEEE Symposium on Logic in Computer Science*, S. 438 – 447. IEEE Computer Society Press, 1994.

[85] Thiagarajan, P. S. und J. G. Henriksen: *Distributed Versions of Linear Time Temporal Logic: A Trace Perspective*. In: Reisig, W. und G. Rozenberg (Hrsg.): *Lectures on Petri Nets I: Basic Modells*, Nr. 1491 in *Lecture Notes in Computer Science*, S. 643 – 679. Springer-Verlag, 1998. Lecture Notes of the 3rd Advanced Course on Petri Nets, Dagstuhl (1996).

[86] Ulrich, A.: *A Description Model to Support Test Suite Derivation for Concurrent Systems*. In: *Kommunikation in verteilten Systemen, GI/ITG-Fachtagung (KiVS'97)*, S. 151–166. Springer-Verlag, 1997.

[87] Ulrich, A.: *Testfallableitung und Testrealisierung in verteilten Systemen*. Shaker Verlag, Aachen, 1998.

[88] Valmari, A.: *State Space Generation: Efficiency and Practicability*. Doktorarbeit, Tampere Univ. of Technology, Tampere, 1988.

[89] Valmari, A.: *Alleviating State Explosion During Verification of Behavioural Equivalence*. Techn. Ber. A-1992-4, Univ. of Helsinki, Dep. of Computer Science, Series of Publications A, 1992.

[90] Valmari, A.: *Stubborn Attack on State Explosion*. Formal Methods in System Design, 1:297–322, 1992.

[91] Valmari, A.: *The State Explosion Problem*. In: Reisig, W. und G. Rozenberg (Hrsg.): *Lectures on Petri Nets II: Applications*, Nr. 1492 in *Lecture Notes in Computer Science*, S. 429 – 528. Springer-Verlag, 1998. Lecture Notes of the 3rd Advanced Course on Petri Nets, Dagstuhl (1996).

[92] Varpaaniemi, K.: *On the Stubborn Set Method in Reduced State Space Generation*. Series A: Research Reports No. 51, Helsinki Univ. of Technology, 1998.

[93] VERNADAT, F., P. AZEMA und F. MICHEL: *Covering step graph*. In: BILLINGTON, J. und W. REISIG (Hrsg.): *17th International Conference on Application and Theory of Petri Nets*, Nr. 1091 in *Lecture Notes in Computer Science*, S. 516–535. Springer-Verlag, 1996.

[94] VERNADAT, F. und F. MICHEL: *Covering step graph preserving failure semantics*. In: P.AZEMA und G.BALBO (Hrsg.): *18th International Conference on Application and Theory of Petri Nets*, Nr. 1248 in *Lecture Notes in Computer Science*, S. 253–270. Springer-Verlag, 1997.

[95] VOGLER, W.: *Modular construction and partial order semantics of Petri nets*. Nr. 625 in *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[96] WALLER, F.: *Model Checking LTL using Net Unfoldings*. In: *Proc of the 10th Conference on Computer Aided Verification*, Nr. 1427 in *Lecture Notes in Computer Science*, 1998.

[97] `xfig` *User's Manual*, 2001. verfügbar unter `http://www.xfig.org/userman`.

[98] ZANDT, T. V.: *PSTricks: PostScript macros for Generic TeX — User's Guide*, 1993.